Universidad Autónoma de Madrid

Escuela Politécnica Superior

TRABAJO FIN DE MÁSTER

Detección de fallos en motores de inducción por medio del uso de técnicas de aprendizaje automático no supervisado

Máster Universitario: Investigación e Innovación en Inteligencia Computacional y Sistemas Interactivos (MU I2-ICSI)

Autor: JARA OVKARIC, Ivo Alonzo

Tutor: FERNÁNDEZ PASCUAL, Ángela

Ponente: DORRONSORO IBERO, José Ramón

3 de septiembre de 2021

DETECCIÓN DE FALLOS EN MOTORES DE INDUCCIÓN POR MEDIO DEL USO DE TÉCNICAS DE APRENDIZAJE AUTOMÁTICO NO SUPERVISADO

Autor: JARA OVKARIC, Ivo Alonzo
Tutor: FERNÁNDEZ PASCUAL, Ángela
Ponente: DORRONSORO IBERO, José Ramón
Departamento: Ingeniería Informática
Centro: Escuela Politécnica Superior

Departamento: Ingeniería Informática
Escuela Politécnica Superior
Universidad Autónoma de Madrid

3 de septiembre de 2021

Agradecimientos

Le agradezco principalmente a mi tutora, Ángela Fernández, por guiarme e indicarme como proceder durante la investigación y desarrollo de este trabajo, corregirme esta memoria y por brindarme apoyo y orientación durante todo el máster.

También le agradezco a José Dorronsoro, por revisar mi investigación y corregir esta memoria. Así como al resto de profesores que han sido parte de mi formación en este máster.

Por último, le agradezco a mi familia y amigos por el apoyo que me han brindado todo este tiempo.

Resumen

Resumen —

En este documento se abordará la aproximación a la detección de fallos en motores de inducción por medio del uso de algoritmos de aprendizaje automático no supervisado. Para esto se analizará la literatura de detección de fallos para encontrar cuales de las señales producidas por los motores de inducción serían las mejores entradas del algoritmo y la calidad de los resultados obtenidos por otros algoritmos de aprendizaje automático no supervisado. Posteriormente, se probará la detección de motores que presenten fallos dentro un conjunto de muestras usando los algoritmos Kernel Principal Component Analysis (KPCA) o Spectral Clustering (SC); siendo el objetivo principal poder separar los motores sanos de aquellos que presentan algún fallo. Al finalizar las pruebas se encontró que con KPCA era posible separar en distintos clusters según el tipo de fallo un subconjunto de los motores disponibles donde se disponía de la misma cantidad de muestras por clase; no obstante, al generalizar a todas las muestras el resultado no fue el esperado. Por otro lado, SC no consiguió buenos resultados ni con el conjunto o subconjunto de muestras. Estos resultados podrán deberse principalmente a que se dispone de muy pocas muestras de motores sanos en comparación con el resto de clases, por lo que a los algoritmos se les dificulta el trabajo de diferenciarlos. A partir de esto se concluye que, si bien no se consiguieron los resultados esperados en este trabajo, es posible que los algoritmos de aprendizaje automático no supervisado puedan detectar fallos en motores de inducción de disponerse de un conjunto de muestras más grande y equilibrado para cada clase de fallo.

Palabras clave — Detección de fallos, Motores de inducción, Kernel Principal Component Analysis, Spectral Clustering.

Abstract

Abstract —

This document will address the use of unsupervised machine learning algorithms to detect faults in induction motors. For this, the fault detection literature will be analyzed to find the best inputs for the algorithm from the signals produced by induction motors and check the quality of the results obtained by other unsupervised machine learning algorithms. Subsequently, some experiments will be run on a set of induction motors signals samples with the main objective of being able to separate healthy motors from those that have a fault using the algorithms Kernel Principal Component Analysis (KPCA) or Spectral Clustering (SC). By the end of the tests it was found that KPCA was able to separate a subset of samples, in which the same amount of samples per class was available, into different clusters by the type of fault that the motor present; however, when generalizing to all the samples, the result was not what was expected. On the other hand, SC did not obtain good results with neither the subset nor the full set of samples. These results may be mainly due to the fact that there are very few samples of healthy motors compared to the rest of the classes, which makes it difficult for the algorithms to differentiate between them. From this work is concluded that, although the obtained results where not the expected, it is possible that unsupervised machine learning algorithms can detect failures in induction motors, especially if it analyzes a well-balanced sample set.

Key words — Fault Detection, Induction motor, Kernel Principal Component Analysis, Spectral Clustering.

Acrónimos

ϵ-neighborhood vecindario ϵ. VI, 11

k-means k-means clustering. VI, 2, 9, 16, 23, 24

BB Barras Rotas. VI, 17, 18, 27

BBSS Barras Rotas y Oscilación Sinusoidal. VI, 17, 18, 27

FFT Transformada Rápida de Fourier. VI, 3, 20, 23, 26

HM Motor Sano. VI, 17, 18, 23, 35, 36

KNN k vecinos próximos. VI, 11

KPCA Kernel Principal Component Analysis. VI, 2, 7, 8, 16, 23, 24, 35

Mutual KNN k vecinos mutuos próximos. VI, 11, 12

PCA Principal Component Analysis. VI, 7

QS Oscilación Cuadrática. VI, 17, 18

SC Spectral Clustering. VI, 2, 10, 11, 16, 23, 27, 35

SOM Self-organizing maps. VI, 1, 4, 5

SS Oscilación Sinusoidal. VI, 17, 18

Índice general

Índice de cuadros

Índice de figuras

Capítulo 1

Introducción

1.1. Introducción

En la industria es pertinente realizar un mantenimiento preventivo de los equipos que se utilizan antes de que sufran una avería que cause la paralización de la actividad en lo que se consigue un repuesto, evitando pérdidas a largo plazo. Para evitar esto se recurre a la detección anticipada de fallos, de manera que a partir de señales producidas por los mismos equipos (corriente, potencia, espectro de frecuencias, etc.) se pueda detectar desperfecto de los mismos. Si bien esta detección la suelen realizar expertos; es posible que por medio de técnicas de inteligencia artificial se pueda automatizar la detección de los fallos. Por lo tanto, se propone poder realizar el mismo trabajo de detección de fallos que se realiza actualmente de forma manual, pero por medio del uso de técnicas de inteligencia artificial que permitan la clasificación y predicción de los mismos [1, 2].

Para esto se parte de lo propuesto en otros trabajos de la literatura de detección de fallos que aplican enfoques similares. Uno de ellos es el trabajo realizado por Bossio [3], donde se concluyó que por medio del uso de algoritmos de inteligencia artificial se podría identificar si los motores de inducción presentan fallos usando los espectros de señales de las corrientes que estos producían. Cabe destacar que en el trabajo realizado por Bossio [3] se disponía de un conjunto de muestras reducido y se consiguió separar los motores sanos de aquellos que presentaban distintos tipos de fallos. Esto sienta las bases de que es posible utilizar técnicas de aprendizaje automático no supervisado para la detección de fallos en motores de inducción. Otro artículo respalda este enfoque es el realizado por Sid et al. [4], donde pudieron separar los motores sanos de aquellos que presentaban barras rotas por medio del algoritmo de aprendizaje automático Self-organizing maps (SOM).

Por lo tanto, el objetivo principal del proyecto será intentar comprobar que es posible detectar distintos fallos en motores de inducción por medio de algoritmos de aprendizaje automático no supervisado a partir de la señal producida por la corriente de los mismos. Principalmente, se buscará separar los motores sanos de aquellos que presentan algún fallo; no obstante, se plantea la posibilidad de poder clasificar los motores con fallos en función de su origen (oscilaciones, barras rotas, etc.) [5]. Así mismo, se analizará si dichos fallos pueden ser detectados utilizando la señal original de la corriente de los motores en lugar del espectro de frecuencias; dado que en caso de conseguirse detectar los fallos solo con la señal de la corriente, se podrá ejecutar el algoritmo en vivo, reduciendo el tiempo de repuesta al fallo.

En cuanto a las técnicas de aprendizaje automático no supervisado, en este trabajo se usarán técnicas de aprendizaje de subvariedades. Esta técnicas se encargan de ubicar las muestras en un espacio de menor dimensionalidad que el espacio original del conjunto de muestras por

medio del uso de kernels no lineales, tal que en la ubicación resultante se vean reflejadas las características propias de las distintas muestras. Las técnicas que se usarán en este trabajo son Kernel Principal Component Analysis (KPCA) y Spectral Clustering (SC). Por un lado, se usará KPCA [6] en conjunto el algoritmo de clustering k-means. KPCA se encargará de ubicar en zonas aledañas los motores que presenten el mismo tipo de fallo y k-means se encargará de distribuir las muestras en distintos clusters a partir de las componentes principales obtenidas. Por otro lado, se utilizará la técnica de SC [7], la cual devuelve como resultado los motores en distintos clusters. Se espera que los clusters obtenidos por ambas aproximaciones puedan agrupar los motores con el mismo tipo de fallo (clase) y que predomine una clase por cluster. Así mismo, basados en la similitud de entre ambos algoritmos según indican [8, 9], se espera que la calidad de los clusters obtenidos por ambos modelos sea similar.

1.2. Objetivos

El objetivo principal de este trabajo es poder detectar, lo más pronto posible, los fallos en motores de inducción por medio del uso de técnicas de aprendizaje automático no supervisado. Para esto, se proponen los siguientes objetivos parciales:

1. Estudiar el estado del arte para el uso de aprendizaje automático en la detección de fallos en motores eléctricos.
2. Estudiar en profundidad métodos de aprendizaje no supervisado de reducción de dimensionalidad.
3. Realizar un buen preprocesamiento de las señales que permita obtener la mayor información posible de las mismas.
4. Obtener un algoritmo que separe los motores sanos de aquellos que presentan algún tipo de fallo.
5. Poder clasificar los tipos de fallos del motor según el origen de los mismos.
6. Detectar los fallos a partir de la señal de la corriente producida por los motores en lugar de usar el espectro de frecuencias.

1.3. Estructura de la memoria

La memoria consta de los siguientes capítulos:

- **Introducción**: en este capítulo se introduce la motivación y enfoque del trabajo de fin de máster, los objetivos a alcanzar al finalizarlo y la estructura de la memoria.
- **Estado del arte**: en este capítulo se expone el estado del campo de la detección de fallos (Fault Detection), especialmente por medio del uso de técnicas de aprendizaje automático; así como los algoritmos de aprendizaje automático que se utilizarán para el desarrollo del trabajo.
- **Experimentos**: en este capítulo se explica con detenimiento los experimentos que se realizaron durante el desarrollo de este trabajo; detallando los datos que se utilizaron, el preprocesamiento que se realizó con ellos, los modelos de aprendizaje automático aplicados en los experimentos y los resultados obtenidos por estos últimos.
- **Conclusiones y Trabajo Futuro**: en este capítulo se exponen las conclusiones obtenidas a partir de la realización del trabajo, comprobando si se han alcanzado los objetivos y proponiendo futuras líneas de trabajo a partir de lo descubierto.

Capítulo 2

Estado del arte

En esta sección se expone el contexto del trabajo, la detección de fallos, así como su relación con las técnicas de aprendizaje automático. En el primer apartado se aborda el estado del arte respecto a la detección de fallos, particularmente en motores de inducción, indicando técnicas para la detección tanto manual como automatizada, entre las que se incluyen distintos enfoques de aprendizaje automático. Por otro lado, en el segundo apartado se revisan los modelos de aprendizaje automático no supervisado que se proponen para poder realizar la detección de fallos, dando un breve repaso de cada modelo.

2.1. Detección de fallos en motores de inducción

El mantenimiento preventivo requiere que se pueda predecir que un equipo va a necesitar mantenimiento sin tener que detener los equipos para comprobar su estado. Para esto es necesario analizar las distintas señales producidas por los motores (corriente, potencia, etc.), así como sus respectivos espectros de frecuencias obtenidos usando la Transformada Rápida de Fourier (FFT), en búsqueda de alguna alteración de su comportamiento habitual [5]. A esto se le conoce como detección de fallos.

Un motor de inducción es un motor de corriente alterna donde el campo magnético de la bobina del estator se encarga de inducir al rotor, por medio de inducción electromagnética, la corriente eléctrica necesaria para producir torsión. Uno de los tipos de fallos que pueden experimentar los motores de inducción es cuando estos presentan oscilaciones producto de desbalances mecánicos. Estas oscilaciones en la velocidad del motor provocan variaciones de las corrientes de alimentación y de la potencia activa entregada al motor, así como fatiga de los elementos de trasmisión como las poleas, correas, rodamientos, etc. Esto se reflejado en el espectro de frecuencias con la aparición de pequeños picos a distintas frecuencias. Otro posible tipo de fallo en los motores de inducción puede deberse a la rotura de una o más barras del rotor. Estas roturas introducen una asimetría eléctrica en los motores debido a la falta de corriente en los mismos, alterando la frecuencia de desplazamiento. Esto se ve reflejado en el espectro de frecuencias como una banda lateral inferior de la corriente fundamental.

Normalmente la detección de fallos la suelen realizar expertos, analizando de forma visual el espectro de frecuencias de las señales de corriente producidas por los motores. No obstante, existe la posibilidad de automatizar este proceso por medio del uso de técnicas de inteligencia artificial para poder realizar el mismo trabajo de detección, clasificación y predicción.

Esta aproximación, utilizar las señales producidas por los motores de inducción como entrada de un algoritmo de inteligencia artificial que sea capaz de clasificar los motores de inducción en grupos según el fallo, es la que se propone en los artículos de Bossio [3] y Sid et al. [4].

En ambos artículos se eligió el algoritmo de aprendizaje automático no supervisado Self-organizing maps (SOM) [10]. El objetivo de este algoritmo es ser capaz de ubicar cada muestra en un mapa autogenerado según las características de las muestras; tal que se formen clusters de motores según el tipo de fallo que presentan. De esta forma, se entrenará al algoritmo con un conjunto de muestras de entrenamiento para luego comprobar que este pueda clasificar correctamente nuevas muestras experimentales. Si bien este algoritmo es diferente de los que se usarán en este trabajo, la propuesta de usar como entradas las señales de las corrientes de los motores de inducción es la misma.

2.1.1. Aplicación de algoritmos de aprendizaje automático para detección de fallos

A continuación se analizará el ejemplo práctico realizado por Bossio [3] donde se aplican algoritmos de aprendizaje automático no supervisado para la detección de fallos en motores de inducción. Con esto se busca analizar cómo se aborda esta aproximación en la literatura y a partir de ello enfocar los experimentos de este trabajo. De esta forma, se busca entender los motivos por los que se elige cierto tipo de señal como entrada del algoritmo, el preprocesamiento que tiene aplicar sobre las mismas, la detección características propias de cada clase de fallo y el resultado que obtiene el algoritmo de aprendizaje automático. Es importante indicar que se eligió este ejemplo para analizar ya que el conjunto de datos con el que trabaja es similar al conjunto de datos que se dispone para este trabajo.

Primero, la entrada que se eligió para los SOM son los espectros de frecuencias de las señales de corriente de los motores de inducción. La elección del espectro de frecuencia sobre la señal de la corriente se fundamenta tanto en el aporte que brinda el propio espectro de frecuencia sobre la actividad periódica de las señales; así como el aporte brindado por los expertos en detección de fallos, quienes revisan el espectro de frecuencia para diferenciar los motores sanos de aquellos que presentan un fallo. Por esto mismo, se realizó un estudio del espectro de frecuencia de las señales de distintos tipos de fallos; del cual se obtuvo información que permite ilustrar las diferencias en el espectro de frecuencia que presenta cada tipo de fallo [3]. En la Figura 2.1 se pueden apreciar los espectros de frecuencia de los motores usados para dicho estudio; a partir de los cuales se puede distinguir claramente las características propias de cada tipo de fallo:

- El motor sin fallos presenta una clara frecuencia predominante a 50 Hz.
- El motor con fallos por oscilación presenta pequeñas frecuencias secundarias entre 40 y 60 Hz, correspondientes a las oscilaciones que interfieren con el motor.
- El motor con barras rotas presenta unos claros picos a 25Hz y 75Hz aproximadamente, correspondientes a excentricidades producto de las barras rotas.

Cabe destacar que en la Figura 2.1 se ha aumentado la resolución de la imagen para centrarse en las diferencias entre los espectros de frecuencia; siendo esto apreciable en el hecho de que no aparece el pico de la frecuencia predominante en ninguna captura. Por este motivo se necesitó aplicar un filtro de paso bajo al espectro de frecuencias, de forma que se descarte el resto de señales que no son pertinentes para el estudio y se pueda centrar más minuciosamente en las diferencias entre los fallos. Así mismo, es de notar que las escalas de las corrientes varían entre los espectros de frecuencias. Esta variación, si bien podría ocasionar que el SOM aprenda a diferenciar los fallos según la escala de la corriente, se ve solventada al normalizar (dividir entre el valor absoluto máximo) cada de las muestras.

Finalmente, luego de aplicar el anterior preprocesamiento a las entradas del SOM, se obtuvieron los resultados mostrados en la Figura 2.2. Como se puede notar en el gráfico, el algoritmo consiguió ubicar las muestras de entrenamiento en clusters según el tipo de fallo

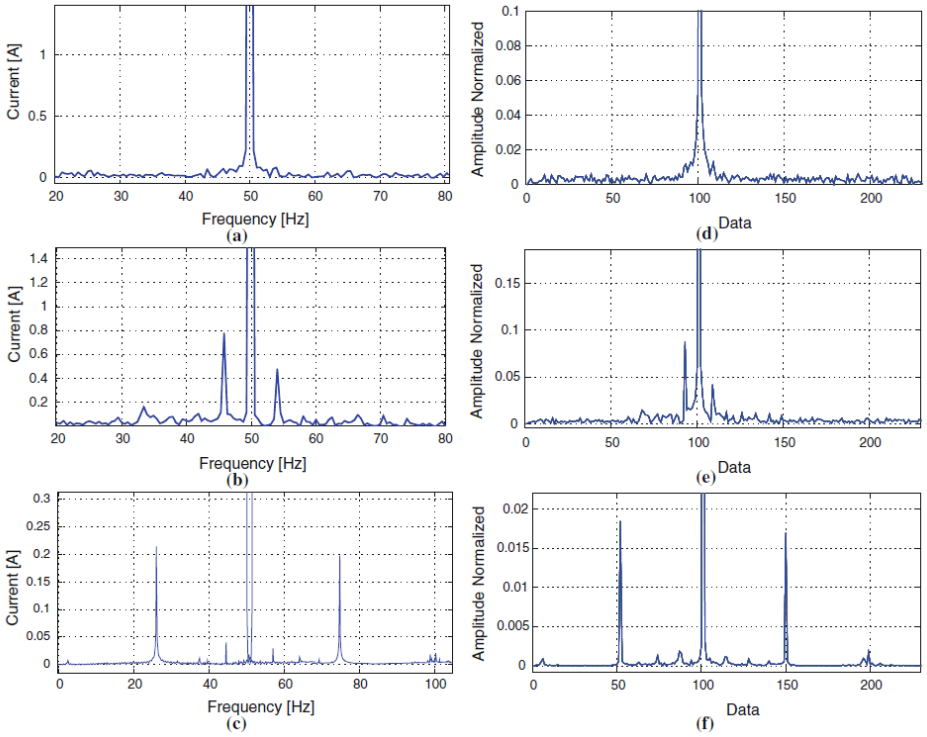


Figura 2.1: Imagen tomada de [3]. La columna de la izquierda corresponde a los espectros de frecuencia medidos en un motor: **a** sin fallos, **b** fallo de oscilación y **c** fallo por barras rotas. La columna de la derecha corresponde a los espectros de frecuencia de los motores que se usaron para la validación: **d** sin fallos, **e** fallo de oscilación y **f** fallo por barras rotas

que presenta el motor. Así mismo, se comprobó que las muestras de validación (D, E y F en la Figura 2.2) se ubicaron correctamente en su respectivo cluster. Cabe destacar que al tener un número limitado de muestras y que las muestras de entrenamiento son simulaciones que parten de las muestras reales de validación, es posible que el resultado del algoritmo presente cierto nivel de sesgo. No obstante, dado que se consiguió un resultado aceptable, se considerará aplicar un preprocesamiento similar en las muestras disponibles para este trabajo.

En resumen, luego de revisar los experimentos realizado por Bossio [3], se ha comprobado que se puede generar un modelo que use algoritmos de aprendizaje automático no supervisado que pueda separar los motores de inducción en función del tipo de fallo que estos presentan correctamente. Así mismo, dicha detectando fallos en los motores se puede realizar a partir del espectro de frecuencias de la señal de corriente que esto producen. Esto se ve reforzado por los resultados conseguidos por Sid et al. [4], donde también ha sido posible distinguir los motores sanos de aquellos que presentan barras rotas usando también un SOM. Por último, dado el correcto desempeño del algoritmo de aprendizaje automático no supervisado SOM, se plantea el uso de otros algoritmos de aprendizaje automático no supervisado como alternativas para detectar fallos en motores de inducción. Dichos técnicas, específicamente las que se utilizarán en este trabajo, se explicarán en el siguiente apartado.

2.2. Algoritmos de aprendizaje automático no supervisado

Como se ha indicado en el apartado anterior, es posible utilizar distintas técnicas de aprendizaje automático para poder detectar, de forma preventiva, fallos en motores de inducción. Debido a que no se conoce a priori la clase de cada muestra se utilizarán técnicas

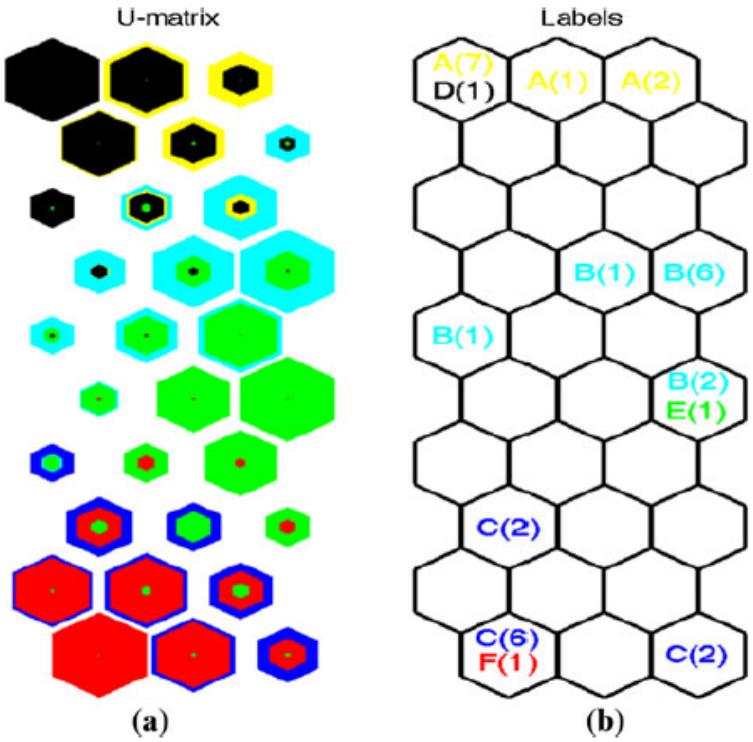


Figura 2.2: Imagen tomada de [3]. La figura de la izquierda corresponde a los histogramas que genera el SOM mientras que la figura de las derecha indica las etiquetas bajo las que clasifica los motores: **A** de prueba con barras rotas, **B** de prueba sin fallos, **C** de prueba con fallos de oscilación, **D** de validación con barras rotas, **E** de validación sin fallos y **F** de validación con fallos de oscilación

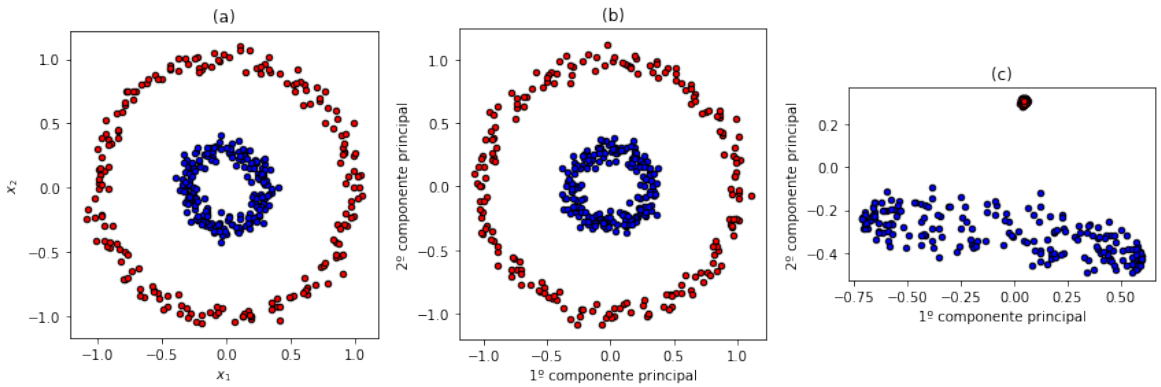


Figura 2.3: Ejemplo de PCA vs KPCA: **(a)** Conjunto de datos original, **(b)** Proyección obtenida por PCA, **(c)** Proyección obtenida por KPCA en el espacio Φ

de aprendizaje no supervisado para poder organizar las muestras. A partir de esto, se plantearon 2 modelos para poder agrupar los motores en distintos grupos (cluster), donde cada grupo se componga de motores sanos o de algún tipo de fallo en específico (motor afectado por oscilaciones o motor con barras rotas). Para esto se utilizaran las técnicas de aprendizaje automático que permitan la reducción de las dimensiones del conjunto de muestras, desplegando las características subyacente del conjunto de muestras y agrupar un conjunto de muestras en k clusters. A continuación se detallarán las técnicas elegidas para este trabajo.

2.2.1. KPCA

Kernel Principal Component Analysis (KPCA) es la versión no lineal Principal Component Analysis (PCA). En PCA se parte de un conjunto de datos X donde

$$x_k \in \mathbb{R}^N, k = 1, ..., l$$

del cual se computa una matriz de covarianzas

$$C = \frac{1}{l} \sum_{j=1}^{l} x_j x_j^T$$

y los autovectores de su proyección ortogonal corresponderán a los componentes principales donde se encontrarán las nuevas muestras.

KPCA se encargará de seleccionar un mapeo ϕ que permitirá mover los datos del espacio $\mathbb{R}^N$ al espacio F, convirtiendo los datos X en $\tilde{X}$, tal que $\tilde{X}$ pueda ser separado linealmente en el espacio F cuando no podía serlo como x en el espacio $\mathbb{R}^N$ [6]. Esto se puede apreciar en la Figura 2.3 donde partiendo del mismo conjuntos de datos **(a)**, PCA no es capaz de separar las clases **(b)** mientras que KPCA si lo es **(c)**.

Para entender como KPCA es capaz de separar linealmente los datos hay que partir de un conjunto de datos X, tal que estos corresponden a $\phi(x_1), ..., \phi(x_l)$ en el espacio F. A partir de estos nuevo dados se forma la matriz de covarianzas

$$\hat{C} = \frac{1}{l} \sum_{j=1}^{l} \phi(x_j)\phi(x_j)^T$$

para la que tienen los autovalores $\lambda \geq 0$ y los autovectores $V \in F \backslash 0$ que cumplan $\lambda V = \hat{C}V$.

Dado que todas las soluciones de V se encuentran en el espacio $\phi(x_1), ..., \phi(x_l)$ se puede obtener la equivalencia

$$\lambda(\phi(x_k)V) = \hat{C}(\phi(x_k)V), k = 1, ..., l$$

además de existir coeficientes (del autovector) $\alpha_1, ..., \alpha_l$ tal que

$$V = \sum_{i=1}^{l} \alpha_i \phi(x_i)$$

A partir de estos se puede definir una matriz K de tamaño $l \times l$ donde:

$$K_{ij} = (\phi(x_i)\phi(x_j))$$

con esto se puede definir

$$l\lambda K\alpha = K^2\alpha$$

lo cual puede satisfacerse resolviendo

$$l\lambda\alpha = K\alpha$$

para todos los autovalores diferentes de 0 donde α es una columna correspondiente a un vector $\alpha_1, ..., \alpha_l$.

Al normalizar α^k para todos los autovalores distintos de 0 es necesario normalizar los vectores V^k del espacio F, tal que se cumpla

$$1 = \sum_{i,j=0}^{l} \alpha_i^k \alpha_j^k (\phi(x_i)\phi(x_j)) = (\alpha^k K \alpha^k) = \lambda_k(\alpha^k \alpha^k)$$

Para la extracción de componentes principales se computa la proyección de la imagen de los puntos $\phi(x)$ en los autovectores V^k del espacio F por medio de

$$(V^k \phi(x)) = \sum_{i=1}^{l} \alpha_i^k (\phi(x_i)\phi(x))$$

donde no es necesario saber el valores de $\phi(x_i)$ dado que se dispone de la matriz K. De esta forma, se puede utilizar el kernel para computar los productos escalares sin necesidad de conocer el mapeo ϕ, siempre que se elija un kernel $k(x, y)$ que ubique los datos en un mapeo ϕ que forme parte del producto escalar del espacio F. Entre esos kernel se encuentran:

- **Kernel polinómico**: $k(x, y) = (x, y)^d$
- **Kernel gaussiano**: $k(x, y) = \exp \frac{-\|x_i - x_j\|^2}{2\sigma^2}$
- **Kernel sinusoidal**: $k(x, y) = \tanh(\kappa(x, y) + \Theta)$

Resumen del algoritmo

En resumen, el algoritmo KPCA sigue los siguientes pasos:

1. Se computa la matriz de productos escalares $K_{ij} = (k(x_i, x_j))_{ij}$.
2. Se resuelve $l\lambda\alpha = K\alpha$ diagonalizando K.
3. Se normaliza los autovectores expandiendo los coeficientes α^n usando:

$$1 = \sum_{i,j=0}^{l} \alpha_i^k \alpha_j^k (\phi(x_i)\phi(x_j)) = (\alpha^k K \alpha^k) = \lambda_k(\alpha^k \alpha^k)$$

4. Se extraen los componentes principales del kernel k al computar la proyección de los autovectores:

$$V^k\phi(x)) = \sum_{i=1}^{l} \alpha_i^k(\phi(x_i)\phi(x)$$

Cabe destacar que en la práctica, si se hace el respectivo mapeo ϕ de los datos, es posible que los resultados obtenidos no sean los mismos que por KPCA, dado que este utiliza el truco del kernel en su lugar.

2.2.2. K-Means Clustering

K-means clustering es un algoritmo de aprendizaje automático no supervisado que identifica k clusters dentro de un conjunto de muestras en un espacio multidimensional [1]. El objetivo de los clusters es minimizar la distancia del centro del cluster con sus puntos (minimizar la varianza del cluster) mientras se maximiza la distancia entre los centros de los distintos clusters.

Para esto se tienen los vectores multidimensionales μ_k donde $k = 1, ..., K$, donde μ_k representa el centro o media del cluster k-ésimo. Partiendo de esto, para cada muestra x_n corresponde un vector r_{nk} donde $r_{nk} \in \{0, 1\}$ tal que si $r_{nk} = 1$ entonces x_n pertenece al cluster k, teniendo $r_{nj} = 0, j \neq k$ ya que solo puede pertenecer a un cluster a la vez.

Por lo tanto, se puede formalizar el siguiente criterio (medida de distorsión) como la suma de los cuadrados de la distancia de cada punto con el centro de su cluster [1]:

$$J = \sum_{n=1}^{N}\sum_{k=1}^{K} r_{nk}\|x_n - \mu_j\|^2$$

Partiendo de esto, se busca encontrar los valores de μ_k y r_{nk} que minimicen J.

Dejando fijo μ_k se tiene que:

$$r_{nk} = \begin{cases} 1 & \text{si } k = argmin_j\|x_n - \mu_j\|^2 \\ 0 & \text{si no} \end{cases}$$

Y con los r_{nk} obtenidos, se deriva J para encontrar los nuevos μ_k, tal que:

$$\mu_k = \frac{\sum_n r_{nk}x_n}{\sum_n r_{nk}}$$

Una vez hecho esto, se repite el proceso iterativo hasta que los centros converjan.

Resumen del algoritmo

En resumen, el algoritmo k-means sigue los siguientes pasos [1, 2]:

1. Se seleccionan las k medias iniciales $\bar{x}_k$, muchas veces de forma aleatoria.
2. Se calculan la distribución de las muestras $r_{nk} = \begin{cases} 1 & \text{si } k = argmin_j\|x_n - \mu_j\|^2 \\ 0 & \text{si no} \end{cases}$ en los clusters.
3. Se calculan los nuevos centros de los clusters $\mu_k = \frac{\sum_n r_{nk}x_n}{\sum_n r_{nk}}$

4. Se repiten los pasos hasta que la distribución converja.

Cabe destacar que en la práctica es posible que k-means no dé el mejor resultado a la primera. Esto es debido a la selección al azar de las medias de partida de los clusters, ya que estas pueden converger en un estado de varianza mínima que no necesariamente agrupe de la mejor forma posible las muestras (un resultado subóptimo [2]). Para esto se recomienda ejecutar varias veces el algoritmo pero con distintas medias de partida para comprobar si el resultado obtenido es, en efecto, la distribución más óptima.

2.2.3. Spectral Clustering

Spectral Clustering (SC) es un algoritmo de aprendizaje automático no supervisado que organiza un conjunto de muestras en un número definido de clusters. A diferencia de otros algoritmos de clustering que buscan que los clusters formados sean lo más compactos posibles en el espacio euclídeo donde se encuentran, Spectral Clustering favorece la conectividad entre los datos. Para esto es necesario construir una matriz/grafo de similitud a partir del conjunto de datos, con la cual se podrá generar un grafo laplaciano. Sobre este grafo laplaciano se realizará un análisis espectral con el cual se podrán obtener autovectores y autovalores; tal que al ordenar los autovalores de menor a mayor sea posible seleccionar los n primeros autovectores y utilizarlos para construir clusters a partir de ellos de forma óptima. [8]

Grafos de similitud

El objetivo de la agrupación (clustering) es que a partir de un conjunto de muestras definidas $(x_1, ..., x_n)$ y una notación de similitud entre las muestras ($s_{ij} \geq 0$) se puedan definir distintos grupos donde las muestras dentro de un mismo grupo sean similares entre sí pero diferentes a las de otros grupos. Una forma de representar la similitud entre las muestras es con el uso de una matriz de similitud $G = (V, E)$. Tal que los vértices v_i corresponden a las muestras x_i y estos vértices estarán conectados por una arista si la similitud s_{ij} entre los mismos es positiva y mayor que un margen definido, ponderándose la arista con un peso de s_{ij} [7]. De esta forma, se abordará el problema de agrupación como el de encontrar las particiones que minimicen la cantidad de aristas y peso de las mismas entre grupos (lo que indica que muestras de los otros grupos no son similares) mientras maximiza la cantidad de aristas y el peso de las mismas dentro de un mismo grupo (demostrando la similitud de las muestras del mismo grupo). Para esto hay que definir las siguientes notaciones:

- $G = (V, E)$ es un grafo no dirigido y ponderado, donde $w_{ij} \geq 0$ representa los vértices del grafo.
- La matriz de adyacencia W donde $W = (w_{ij})_{i,j=1,...,n}$. y $w_{ij} = 0$ indica que los vértices v_i y v_j no están conectados.
- La matriz diagonal D es la matriz de grado (Degree matrix), cuya diagonal representa los grados $d_1, ..., d_n$.
- El subconjunto de vértices $A \subset V$, tal que$\mathbb{1}_A = (f_1, ..., f_n)'$ y $f_i = 1$ si $v_i \in A$ si no $f_i = 0$.
- Para cada $A \subset V$: $|A|$ es el número de vértices de A y $vol(A) = \sum_{i \in A} d_i$.

Tipos de grafos de similitud

Existen distintas opciones de grafos de similitud que se pueden elegir para el algoritmo. A continuación se detallan los 3 principales tipos de grafos de similitud recomendados en Spectral Clustering [7]:

- **Grafo de vecindario ϵ (ϵ-neighborhood)** : $w_{ij} = \begin{cases} 1 & \text{si } \|x_i - x_j\| \leq \epsilon \\ 0 & \text{si no} \end{cases}$
- **Grafo de k vecinos próximos (KNN)**: Se crean las aristas entre v_i y los k vértices más similares a v_i. Este puede ser simple, en este caso se ignora la dirección de las aristas y basta con que v_i o v_j se encuentren entre los K-vecinos del otro; o mutuo, en este caso es necesario que ambos nodos estén conectados (ida y vuelta), es decir, tanto v_i como v_j deben estar dentro del k vecindario del otro.
- **Grafo completamente conexo**: Se construyen aristas entre todos los vértices del grafo y se les asigna un peso s_{ij} utilizando una función de similitud que represente las características/variedades locales de los datos tal que los puntos que se consideren similares si estén en efecto relacionados. Un ejemplo sería la función (kernel) de similitud Gaussiana: $s(x_i, x_j) = \exp \frac{-\|x_i - x_j\|^2}{2\sigma^2}$ donde σ controla el tamaño de los vecindarios de forma similar a ϵ en el Grafo de vecindario ϵ; siendo esta la función que se usa por defecto para muestras en el espacio euclídeo $\mathbb{R}^d$.

Selección de parámetros

Como se puede apreciar, cada tipo de grafo de similitud cuenta con distintos parámetros que afectan el desempeño del grafo y por consiguiente el resultado obtenido por SC. A continuación se mencionan recomendaciones a tener en consideración al momento de seleccionar los valores de dichos parámetros [7]:

- **Grafo de vecindario ϵ (ϵ-neighborhood)** : Se debe seleccionar el tamaño de la arista más larga del árbol minimal generado a partir del grafo completamente conexo como ϵ para obtener el menor ϵ posible que permita un grafo conexo.
- **Grafo de k vecinos próximos (KNN)**: Se recomienda que k sea del orden de $\log(n)$ para conjuntos de muestras grandes y para conjuntos pequeños, el orden no es relevante y se fija por prueba y error.
- **Grafo de k vecinos mutuos próximos (Mutual KNN)**: Se recomienda un k mayor al que se elegiría para KNN, ya que a veces es necesario "permitir"la unión de componentes no conexas que pertenecen a la misma clase. No obstante, no hay heurística definida.
- **Grafo completamente conexo**: Una opción es elegir un σ en el orden de la distancia media de un punto a sus k vecinos, siendo $k \sim \log(n) + 1$. Otra opción es seleccionar el σ de forma similar a la selección del ϵ para el vecindario ϵ

Ventajas y desventajas de cada tipo de grafo de similitud

Dadas las notables diferencias entre las posibles opciones de grafo de similitud, a continuación se detallan las principales ventajas y desventajas de cada uno [7]:

- **Grafo de vecindario ϵ (ϵ-neighborhood)** : Es complicado elegir el ϵ ya que puede haber clases que se encuentren en zonas densas del grafo (que estén altamente conectadas) pero que otras clases del mismo grafo se encuentren más dispersas (menos conexiones en comparación a las otras clases).
- **Grafo de k vecinos próximos (KNN)**: Puede conectar clases indiferentemente de la densidad de la zona del grafo, dado que considera los puntos próximos no el tamaño de sus caminos. No obstante, esto puede ocasionar que se conecten componentes de clases distintas debido a su proximidad en el grafo (un par de puntos son lo suficientemente similares como para considerarse ambas particiones como parte del mismo cluster).

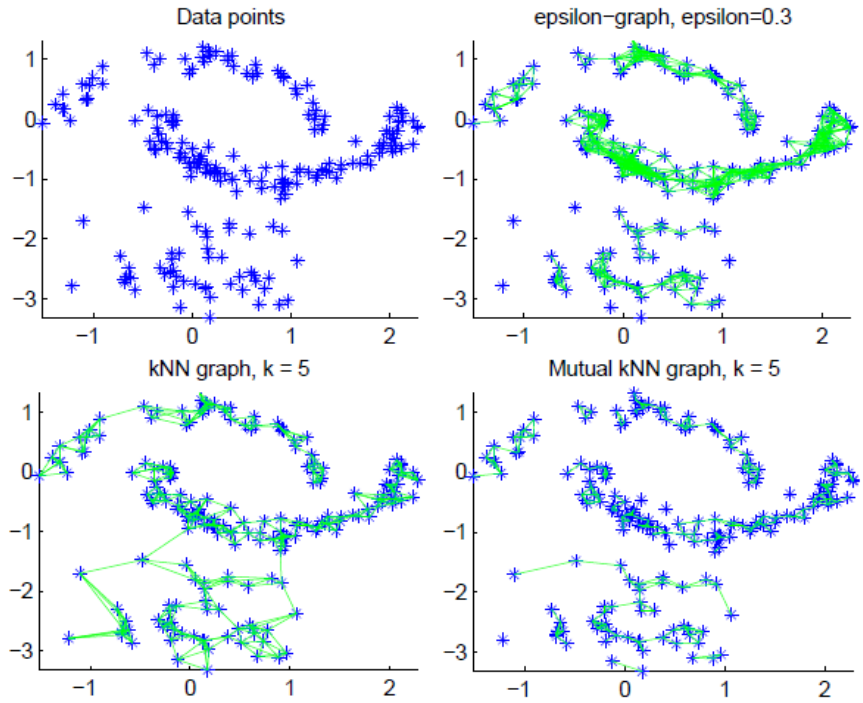


Figura 2.4: Ejemplo de la influencia del tipo de grafo de similitud tomado de U. von Luxburg, "A tutorial on spectral clustering", 2007 [7].

- **Grafo de k vecinos mutuos próximos (Mutual KNN)** : Es un punto medio entre KNN y vecindario ϵ, conecta áreas de densidades parecidas pero evitando conectar con áreas de diferentes densidad. En otras palabras, las conexiones locales provocarán que los vértices de otras particiones se queden fuera de los k-vecinos de los puntos fronterizos.
- **Grafo completamente conexo**: De forma similar al ϵ, el tamaño del σ afecta la conectividad ya que los vecindarios locales están conectados por similitudes altas mientras que las conexiones con otros puntos lejanos son > 0 pero desdeñables.

Estas diferencias se puede apreciar en la Figura 2.4, donde el resultado final del algoritmo varía en función del grafo de similitud elegido.

Grafos Laplacianos

La matriz Laplaciana es una matriz definida positiva que parte de la matriz de pesos W y se usa para encontrar los autovalores y autovectores con los que se harán, posteriormente, la agrupación de las muestras. Para definir una matriz laplaciana se considera [7]:

- G es un grafo ponderado no dirigido.
- W es la matriz de pesos de G, donde $w_{ij} = w_{ji} \geq 0$.
- Los autovectores de la matriz no estarán necesariamente normalizados. Ej: el vector constante $\mathbb{1}$ y a que multiplica $a\mathbb{1}$ para algún $a \neq 0$ será considerado como el mismo autovector.
- Los autovalores se ordenarán del menor al mayor.

A continuación se describen diferentes tipos de matrices Laplacianas.

Grafo Laplaciano sin normalizar

Se define como $L = D - W$, donde W es la matriz de peso (similitud) y D la matriz de grado (Degree matrix). Esta matriz presenta las siguientes características:

- Para cada vector $f \in \mathbb{R}^n$ se cumple: $f^{'}Lf = \frac{1}{2}\sum_{i,j=1}^{n} w_{ij}(f_i - f_j)^2$.
- Es simétrica y semidefinida positiva ($f^{'}Lf \geq 0$).
- El autovalor más pequeño es 0 y corresponde al vector constante $\mathbb{1}$.
- Tiene n autovalores reales no negativos $0 = \lambda_1 \leq \lambda_2 \leq ... \leq \lambda_n$

De esta forma, dado un grafo G no dirigido y con pesos no negativos, se tiene que la cantidad k de autovalores de L iguales a 0 corresponde a la misma cantidad de componentes conexas $A_1, ..., A_k$ del grafo. Así mismo, los autovalores 0 son generados por las componentes $\mathbb{1}_1, ..., \mathbb{1}_k$. Para probar esto se parte del caso k=1, el caso del grafo conexo, se asume que f es un autovector con autovalor 0; tal que

$$0 = f^{'}Lf = \sum_{i,j=1}^{n} w_{ij}(f_i - f_j)^2$$

como w_{ij} no es negativo y existe al menos una arista entre 2 vértices, se debe cumplir que $(f_i - f_j)^2$, por lo tanto $f_i = f_j$ y f tiene que ser constante. Como la única componente constante es $\mathbb{1}$, entonces el autovector f tiene que ser $\mathbb{1}$ con su respectivo autovalor 0. Confirmando así que los autovalores 0 indican las componentes conexas.

Esto se puede extrapolar para k componentes conexas, al considerar a L como la unión de los espectros $L_i, 1 \leq i \leq k$. Cada L_i tendrá un autovalor 0 para el $\mathbb{1}$, por lo que la matriz L tendrá tantos autovalores 0 como componentes conexas tenga.

Grafo Laplaciano normalizado

Existen 2 matrices laplacianas normalizadas:

- **Laplaciano de matriz simétrica**: $L_{sym} = D^{-\frac{1}{2}}LD^{\frac{1}{2}} = I - D^{-\frac{1}{2}}WD^{\frac{1}{2}}$
- **Laplaciano de matriz camino aleatorio**$L_{rw} = D^{-1}L = I - D^{-1}W$

Las matrices laplacianas normalizadas presentan las siguientes características:

- Para cada vector $f \in \mathbb{R}^n$ se cumple: $f^{'}L_{sym}f = \frac{1}{2}\sum_{i,j=1}^{n} w_{ij}(\frac{f_i}{\sqrt{d_i}} - \frac{f_j}{\sqrt{d_j}})^2$.
- λ es autovalor del autovector u de L_{rw} si y solo si λ es autovalor del autovector w de L_{sym} y $w = D^{-\frac{1}{2}}u$
- λ es autovalor del autovector u de L_{rw} si y solo si λ y u resuelven $Lu = \lambda Du$
- L_{sym} y L_{rw} son semidefinidas positivas ($f^{'}Lf \geq 0$).
- L_{sym} y L_{rw} tienen n autovalores reales no negativos $0 = \lambda_1 \leq \lambda_2 \leq ... \leq \lambda_n$
- El autovalor más pequeño de L_{rw} es 0 y corresponde al vector constante $\mathbb{1}$.
- El autovalor más pequeño de L_{sym} es 0 y corresponde al vector constante $D^{-\frac{1}{2}}\mathbb{1}$.

De forma similar, con el laplaciano sin normalizar se cumple que la cantidad k de autovalores 0 corresponde a la cantidad de componentes conexas. Tal que:

- Para L_{rw} los autovalores 0 son generados por las componentes $\mathbb{1}_{A_i}$.
- Para L_{sym} los autovalores 0 son generados por las componentes $D^{\frac{1}{2}}\mathbb{1}_{A_i}$.

Elección del laplaciano

Se recomienda utilizar el laplaciano normalizado L_{rw} y L_{sym} en lugar del laplaciano sin normalizar L. A continuación de describen algunas similitudes y diferencias que favorecen a laplacianos normalizados [7]:

- Ambos laplacianos son capaces de encontrar una partición de puntos que minimice la similitud de las muestras entre distintos clusters (minimizar el corte $cut(A, \hat{A})$).
- Los laplacianos normalizados buscan encontrar una partición que maximice la similitud entre las muestras del mismo cluster (maximizar las similitudes $W(A, A)$ y $W(\hat{A}, \hat{A})$) mientras que el laplaciano sin normalizar no tiene en consideración esto.
- Para grafos de similitud completamente conexos, el laplaciano sin normalizar tiende a presentar problemas en la convergencia de los autovectores en función del sigma que se seleccione. En otras palabras, la elección del sigma puede propiciar que los autovectores no converjan o lo hagan a soluciones triviales (función delta de Dirac) antes de que estos logren explicar el espacio de datos originales. Por otro lado, los laplacianos normalizados no presentan este problema, ya que son capaces de converger a un espacio embebido $C(X)$ y se puede interpretar por los autovectores U de forma similar a la interpretacion del camino aleatorio.

Por último, la diferencia entre los 2 laplacianos normalizados, es que los autovectores de L_{rw} son indicadores de los vectores $\mathbb{1}_{A_i}$ mientras que los de L_{sym} son de $D^{\frac{1}{2}}\mathbb{1}_{A_i}$.

Diferencias producidas por la elección del tipo de grafos de similitud y laplaciano

En la Figura 2.5 se muestra cómo la elección del tipo de grafo de similitud y del laplaciano genera distintos autovalores y autovectores para el mismo conjunto de datos. El conjunto de datos está compuesto por 200 muestras producto de la mezcla de 4 gaussianas. En la primera fila se usó L_{rw} y KNN, en la segunda L y KNN, en la tercera L_{rw} y grafo completamente conexo con kernel gaussiano y $\sigma = 1$ y en la cuarta L y grafo completamente conexo con kernel gaussiano y $\sigma = 1$.

Camino Aleatorio

Un camino aleatorio es un proceso estocástico por el cual se salta de un vértice a otro en el grafo al azar [7]. A partir de esto se puede interpretar Spectral Clustering como tratar de encontrar una partición del grafo que cause que los saltos aleatorios se den principalmente a vértices del mismo cluster y raramente salte a vértices fuera del mismo. De esta forma, una partición balanceada tendrá pocas opciones de recorrer un camino que lo saque del cluster (hay pocas aristas entre los clusters). De esta forma, usando el camino aleatorio se tiene que:

- La probabilidad de saltar de un vértice a otro viene dado por $p_{ij} = \frac{w_{ij}}{d_i}$.
- Se puede definir la matriz de transición como $P = D^{-1}W$.
- Para un grafo conexo no bipartito, el camino aleatorio sigue una distribución estacionaria $\pi = (\pi_1, ..., \pi_n)'$
- Se puede definir L_{rw} en función de P, $L_{rw} = I - P$.
- λ es autovalor del autovector u de L_{rw} si y solo si $1 - \lambda$ es autovalor del autovector u de P.
- Se pueden usar los autovectores más grandes de P y los autovectores más pequeños de L_{rw} para describir las propiedades de los clusters del grafo.

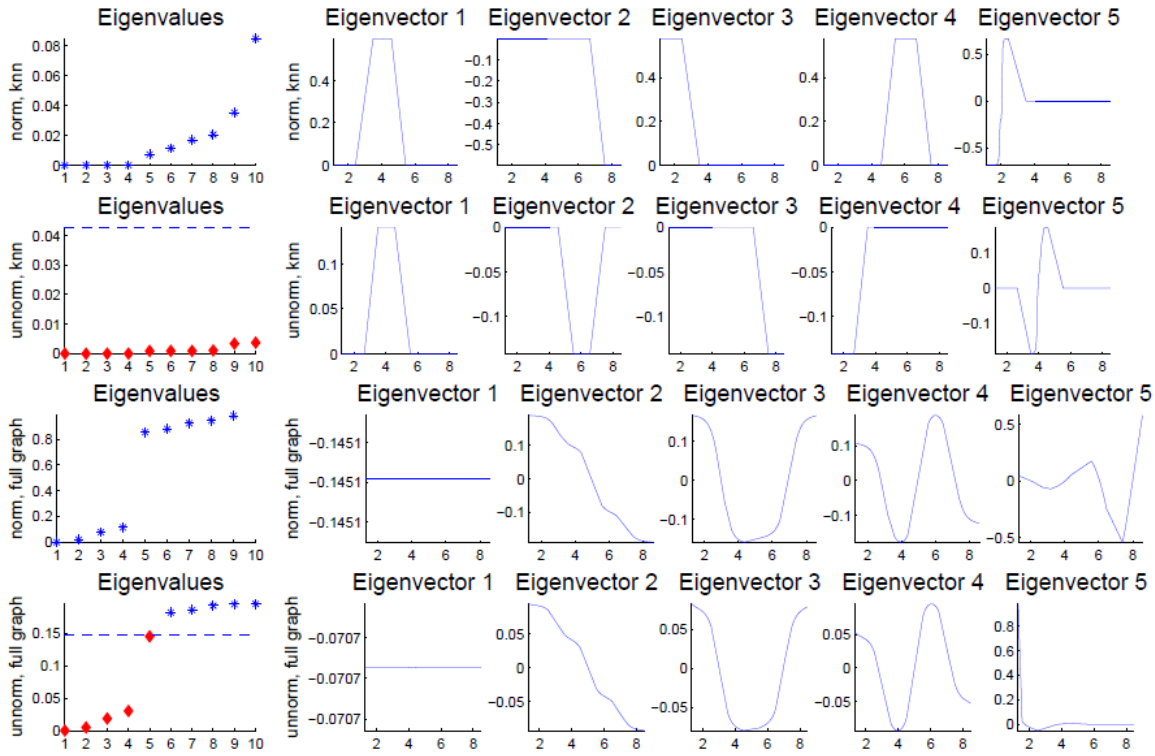


Figura 2.5: Ejemplo de la influencia elección del tipo de grafo de similitud y del lapaciano en Spectral Clustering tomado de U. von Luxburg, "A tutorial on spectral clustering", 2007 [7].

Distancias de conmutación

Una conexión entre el camino aleatorio y el grafo laplaciano es mediante la distancia de conmutación del grafo. La distancia de conmutación c_{ij} es el tiempo esperado que le toma al camino aleatorio para ir de v_i a v_j [7]. Entre la propiedades de la distancia de conmutación está que cuanto mayor sea la cantidad de caminos cortos entre v_i y v_j, menor será c_{ij}. Esto significa que puntos conectados por caminos cortos en áreas de la misma densidad del grafo serán considerados más similares a otros puntos conectados por caminos cortos pero en áreas de distintas densidades, obteniéndose resultados interesantes para hacer clusters.

$$c_{ij} = vol(V)(e_i - e_j)' L^{\dagger}(e_i - e_j)$$

donde $e_i = (0, 0, ..., 0, 1, 0, ..., 0)'$ es el i-ésimo vector unitario y $L^{\dagger}$ es la inversa generalizada del grafo Laplaciano L.

De esto se obtiene que $\sqrt{c_{ij}}$ puede considerarse como la distancia euclídea entre los vértices del grafo. Siendo que así se podría construir un espacio embebido donde se ubique los vértices v_i en puntos $z_i \in \mathbb{R}$ cuyas distancias euclídeas correspondan a las distancias de conmutación; haciendo un símil a cómo Spectral Clustering ubica en el espacio embebido U los puntos y_i.

Elección del número de clusters

Una buena forma de elegir el número de clusters es tener en consideración la heurística de los autoespacios (eigengaps) del laplaciano. Sean los k primeros autovalores $\lambda_1, ..., \lambda_k$ y el k+1-ésimo autovalor λ_{k+1} se busca encontrar un k donde $|\lambda_{k+1} - \lambda_k|$ sea muchísimo mayor a las diferencias entre los k primeros autovalores y sus respectivos predecesores, estando estas más cercanas a 0. En otras palabras, se busca encontrar un k que muestre un gran salto entre el tamaño de los autovalores, siendo este k la cantidad tentativa de clusters que se pueden inferir

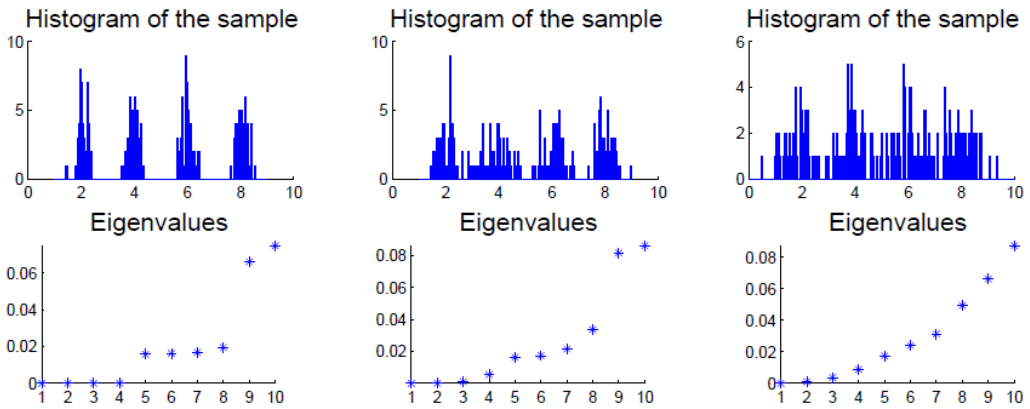


Figura 2.6: Ejemplo de como la distancia entre los autovalores puede sugerir la cantidad de clusters a formar con Spectral Clustering tomado de U. von Luxburg, "A tutorial on spectral clustering", 2007 [7].

a partir del laplaciano, como se puede mostrar en la Figura 2.6. No obstante, esta solución se encuentra muy ligada a la elección de la matriz de similitud y sus respectivos parámetros. También hay que considerar que el escalado de los datos puede afectar la formación de los clusters.

Relación con k-means clustering

Normalmente se suele usar k-means como paso final para obtener los clusters a partir del espacio embebido generado por el laplaciano. No obstante, no es del todo necesario usar este paso, dado que es posible que el mismo laplaciano logre agrupar claramente las muestras en clusters, aunque este caso tiende a ser más cercano a lo ideal que a la realidad. No obstante, cualquier otro algoritmo de clustering puede reemplazar el paso de k-means; así como tener en consideración las distancias euclídeas entre los puntos una vez estos se encuentren embebidos en el espacio euclídeo final.

Relación con Kernel Principal Component Analysis

Así mismo, según lo observado en [8, 9] los algoritmos de Spectral Clustering guardan similitudes con KPCA. En comparación con un KPCA que usa un kernel gaussiano, la principal diferencia se encuentra en la normalización de las matrices. Si bien estas normalizaciones mejoran el desempeño de SC, bajo las condiciones correctas es posible que KPCA obtenga un mejor clustering que Spectral Clustering y viceversa.

Capítulo 3

Experimentos

En esta sección se detallarán los diferentes experimentos que se han realizado para comprobar las hipótesis de detección de fallos planteadas. A continuación se detallará las características del conjunto de datos de señales de motores de inducción del que se dispone, los preprocesamientos que se aplicarán a dichas señales, la metodología de los experimentos planteados, incluyendo las etapas de cada prueba, y los resultados obtenidos de dichas pruebas.

3.1. Conjunto de datos

Para este trabajo se dispone de una colección de 151 señales correspondiente a la corriente producida por motores de inducción que presentan distintos fallos. Como se mencionó en el capítulo anterior, en los motores de inducción (MI) los fallos se ocasionan debido a la presencia de oscilaciones de baja frecuencia o la ruptura de los rotores (barras) del motor. Para los experimentos se disponen de más patrones de fallos que los descritos en el artículo de Bossio [3]. En concreto, aparte de los casos de motor sano y motor con barras rotas, hay motores que presentan oscilaciones sinusoidales, otros oscilaciones cuadráticas y otros que presentan barras rotas y oscilaciones sinusoidales al mismo tiempo. Las distintas clases de fallos, así como el número de muestras por cada clase se pueden apreciar en el Cuadro 3.1. Por otro lado, todas las señales tienen una longitud de 128000 muestras y una frecuencia de muestreo de 8000 muestras/segundo.

Fallo (Clase)	Abreviatura	Nºcasos
Motor Sano (Sin fallos)	HM	10
Oscilación Sinusoidal	SS	40
Oscilación Cuadrática	QS	40
Barras Rotas	BB	30
Barras Rotas y Oscilación Sinusoidal	BBSS	31
Total		151

Cuadro 3.1: Nº de casos disponibles por cada tipo de fallo

Así mismo, los patrones se pueden clasificar por el número de barras rotas. De esta forma se tiene la distribución que indica el Cuadro 3.2. Cabe destacar que el claro desequilibrio entre los casos 0 barras rotas y 1 o más barras rotas corresponde a que los primeros corresponde las clases Motor Sano (HM), Oscilación Sinusoidal (SS) y Oscilación Cuadrática (QS); mientras que el resto corresponden a las clases Barras Rotas (BB) y Barras Rotas y Oscilación

Nº Barras Rotas	NºCasos
0	90
1	20
2	20
3	21
Total	151

Cuadro 3.2: Nº de casos disponibles por Nº de barras rotas.

Sinusoidal (BBSS).

De la clases anteriormente mencionadas, hay una clase que corresponde a motores sin fallos (HM) y 4 clases que corresponden a motores con fallos (SS, QS, BB y BBSS). Por lo tanto, se buscará que los modelos puedan separar principalmente en clusters las clases sin fallos de las clases con fallos; considerándose especialmente bueno si también puede separar las clases que corresponden a motores con fallos entre sí.

Como se puede apreciar en el Cuadro 3.1, hay un claro desequilibrio entre cada clase de fallo de motor de inducción. Esto es más evidente al notar que solo se disponen de 10 muestras de la clase HM (motor sin fallo), frente a 141 muestras de motores con algún tipo de fallo (oscilación, barras rotas, ambos, etc.). Debido a esto, se ha considerado prudente el equilibrar la distribución de las muestras; por lo que se han seleccionado 10 muestras al azar de cada clase, formando el conjunto de pruebas que se muestra en el Cuadro 3.3.

Fallo (Clase)	Abreviatura	NºCasos
Motor Sano (Sin fallos)	HM	10
Oscilación Sinusoidal	SS	10
Oscilación Cuadrática	QS	10
Barras Rotas	BB	10
Barras Rotas y Oscilación Sinusoidal	BBSS	10
Total		50

Cuadro 3.3: Nº de muestras por cada tipo de fallo en el conjunto de pruebas básicas

De esta forma, se disponen de 2 conjuntos de muestras con las que realizar los experimentos:

- El conjunto de muestras original, con 151 muestras y las clases no equilibradas.
- El conjunto de pruebas básicas, con 50 muestras y las clases equilibradas.

Por último, cabe mencionar que las señales de las que se dispone corresponden a motores de inducción con distintos niveles de carga al momento de tomarse las muestras. En el Cuadro 3.4 se muestra cuántas muestras de cada nivel de carga hay. Habrá que tener en consideración esta característica al momento de realizar el preprocesamiento de las señales ya que se debe evitar que los algoritmos organicen las muestras según el nivel de carga. Esto es debido a que este un dato que se conoce a priori, por lo que no aporta nueva información.

3.2. Preprocesamiento de los datos

En este apartado se analizará el preprocesamiento que se deberá aplicar a las señales de la corriente de los motores de inducción para que se puedan usar como entrada de los algoritmos de aprendizaje automático. Para esto se analizarán las características de las señales de los

Nivel de carga (%)	NºCasos
00	30
20	31
40	30
60	30
80	30
Total	151

Cuadro 3.4: Nº de casos disponibles por nivel de carga del motor.

distintos tipos de motor, encontrando las mejores formas de preprocesarlas y que a su vez proporcionen información relevante.

En la Figura 3.1 se encuentran representadas 8 señales de corriente correspondientes a diferentes casos del conjunto de muestras original. Como se puede apreciar a simple vista, hay diferencias claras entre las señales que presentan diferentes fallos. Por ejemplo, se puede notar el efecto de las oscilaciones en las señales así como las ligeras variaciones en la amplitud producidas por las barras rotas cada cierto tiempo. Si bien en un principio se podrían generalizar los aspectos de las señales, calculando una media y variación a partir de estos ejemplos con la cual poder clasificar, es necesario aclarar que la amplitud de las corrientes varía incluso dentro de la misma categoría de fallo. Esto puede deberse principalmente a que, a pesar de pertenecer a la misma categoría de fallo, cada muestra corresponde a un diferente nivel de carga.

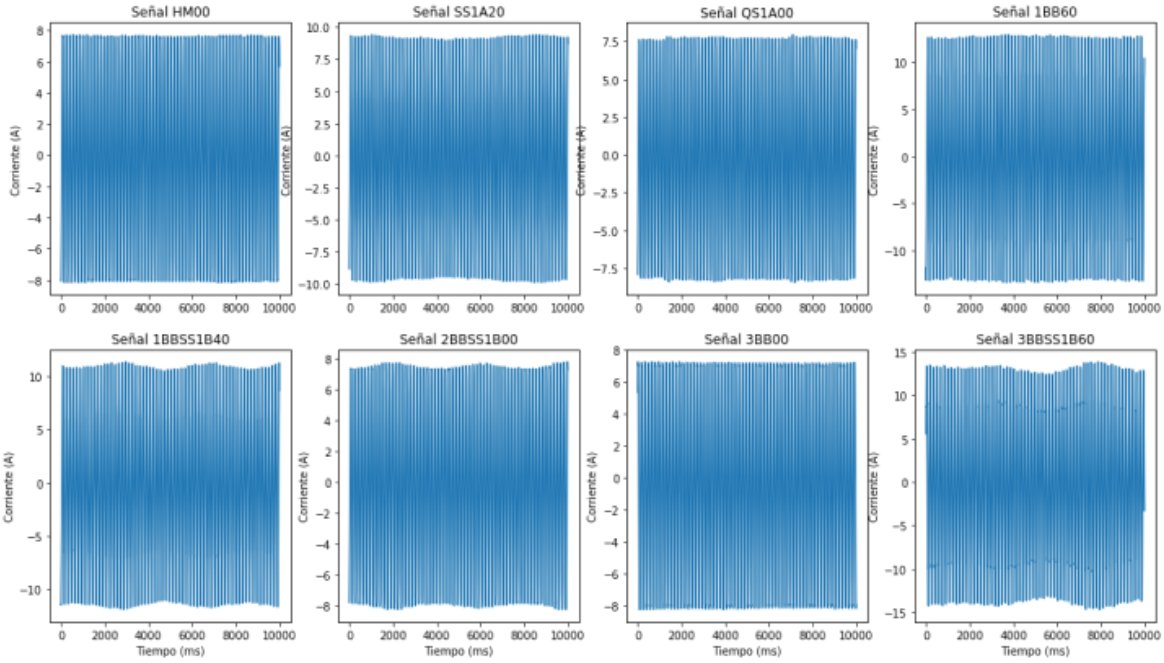


Figura 3.1: Señales de corriente originales producidas por los motores de inducción (10 segundos). **HM** Motor sano, **SS** Motor afectado por oscilación sinusoidal, **QS** Motor afectado por oscilación cuadrática, **xBB** Motor afectado por x (de 1 a 3) barras rotas y **xBBSS** Motor afectado por oscilación sinusoidal y por x barras rotas.

Una primera aproximación para preprocesar los datos sería tomar el valor absoluto de las señales originales. No obstante, esta aproximación sigue siendo susceptible a los cambios en las amplitudes de la señal producto de los distintos niveles de cargas; por lo que a priori

se espera que los algoritmos de aprendizaje automático que utilicen esta entrada terminen organizando los datos por nivel de carga. Si bien en principio este resultado sería correcto, ya que se lograría una separación en clusters en función del nivel de carga con el que funciona cada motor de inducción, no sería relevante ya que este es un dato conocido y el modelo no estaría aportando nueva información. Por este motivo, para evitar que se clasifiquen las muestras por niveles de cargas, el primer paso del preprocesamiento de datos es reducir la varianza en las amplitudes de la muestras.

Para reducir la varianza en las amplitudes de las muestras se han propuesto 2 opciones: normalizar la muestras y escalar las muestras. El normalizar las muestras sigue lo propuesto en el artículo de Bossio [3] y consiste en dividir cada muestra entre su valor absoluto máximo de manera que la señal normalizada se encontrará entre 1 y -1. La señal normalizada correspondiente a las 8 señales mostradas en la Figura 3.1 se encuentra en la Figura 3.2, donde se puede ver cómo se han ajustado las amplitudes sin perder la forma de las señales. Por otro lado, el escalar las señales se encarga de transformar los valores de la señal para que estos queden con una media de 0 y desviación de 1. La señal escalada correspondiente a las 8 señales mostradas en Figura 3.1 se encuentra en la Figura 3.3, donde se puede identificar que, si bien las señales han perdido sus formas originales, estas siguen teniendo formas que distinguen entre diferentes tipos de fallo. En este trabajo, se probarán ambas aproximaciones y se comprobará al final cuál obtiene mejores resultados.

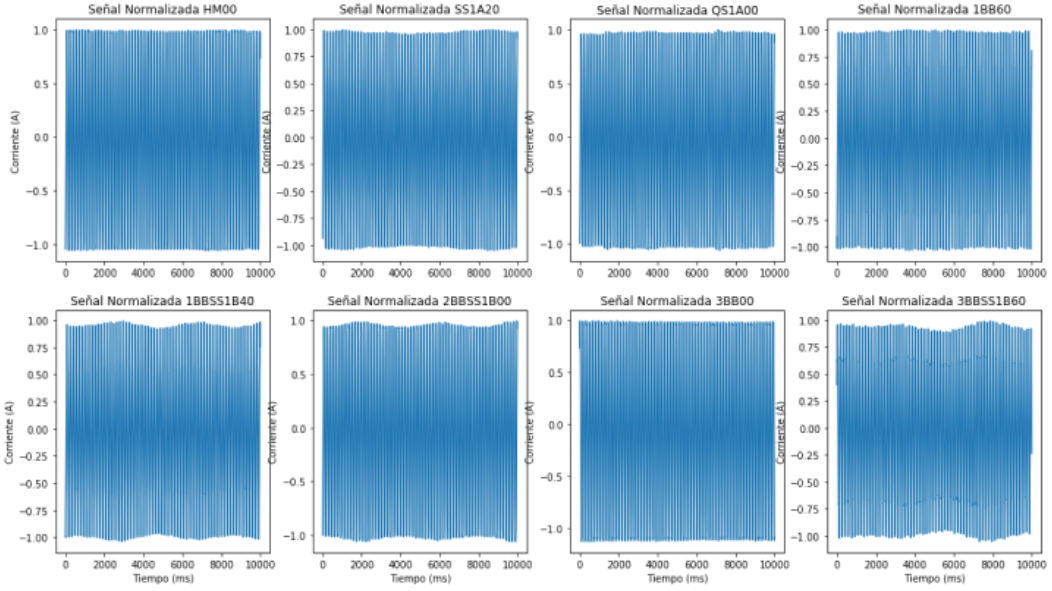


Figura 3.2: Señales de corriente normalizadas (10 segundos). **HM** Motor sano, **SS** Motor afectado por oscilación sinusoidal, **QS** Motor afectado por oscilación cuadrática, **xBB** Motor afectado por x (de 1 a 3) barras rotas y **xBBSS** Motor afectado por oscilación sinusoidal y por x barras rotas.

Una vez normalizadas/escaladas las señales el siguiente paso es obtener los espectros de frecuencias de señales de las muestras, aplicando la Transformada Rápida de Fourier (FFT). De forma similar a lo mostrado en el artículo de Bossio [3], los picos de frecuencias de las señales se encuentra a valores bajos, como se pueden ver en las Figuras 3.4 y 3.5. Entonces, para obtener la información con la frecuencia que muestra los detalles que diferencian los espectro de señales de cada tipo de fallo entre sí es necesario hacer un filtro de paso bajo. Según lo propuesto por Bossio [3], el filtro que se ha aplicado consiste en quedarse todas las señales por debajo de 200 Hz, obteniendo las frecuencias de las Figuras 3.6 y 3.7. Cabe destacar que de forma similar a lo propuesto en el artículo de Bossio [3] en este trabajo se

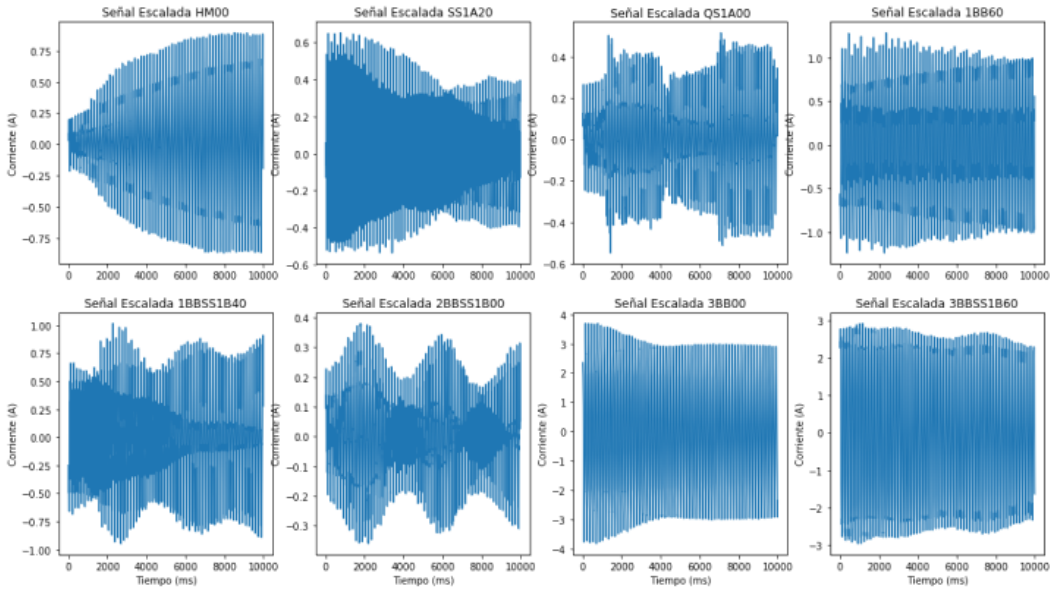


Figura 3.3: Señales de corriente escaladas (10 segundos). **HM** Motor sano, **SS** Motor afectado por oscilación sinusoidal, **QS** Motor afectado por oscilación cuadrática, **xBB** Motor afectado por x (de 1 a 3) barras rotas y **xBBSS** Motor afectado por oscilación sinusoidal y por x barras rotas.

ha aplicado un acercamiento a los espectros de frecuencias para poder identificar los detalles que las diferencian entre sí. De esta forma, se pueden notar los pequeños picos de frecuencia correspondientes a las oscilaciones así como los picos dispares producidos por las barras rotas. Además, se puede apreciar que en caso del espectro de frecuencias del motor afectado por la oscilación cuadrática este no es tan suavizado como el resto de casos; siendo este fenómeno mucho más claro en la Figura 3.6, correspondiente a la señal normalizada, por lo que a priori se puede prever que KPCA podrá distinguir mejor este caso del resto de señales. Por último, también se ha considerado el filtrar a 100 Hz ya que según se aprecia en las Figuras 3.6 y 3.7 la frecuencia principal de la señal parece encontrarse en dicho orden.

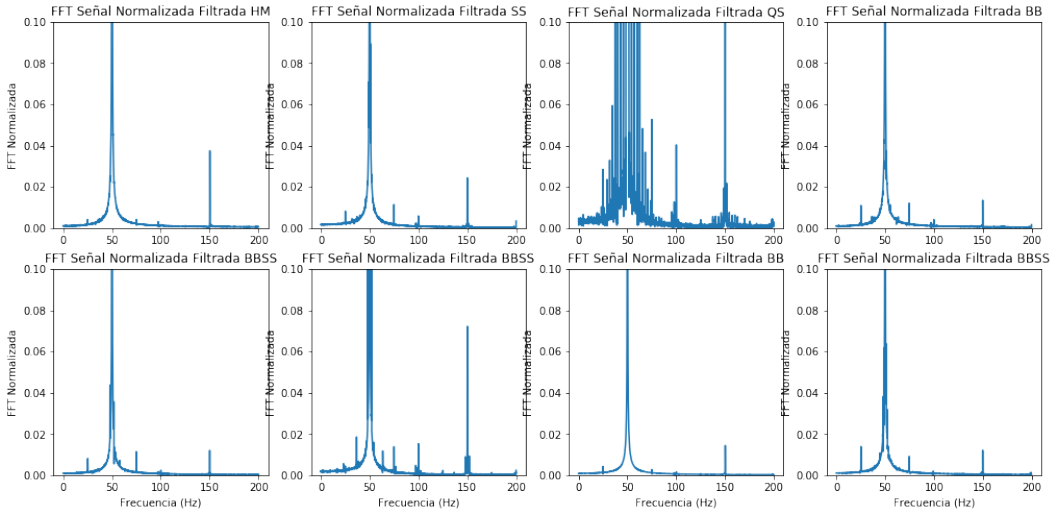


Figura 3.6: FFT filtrada de la señal normalizada

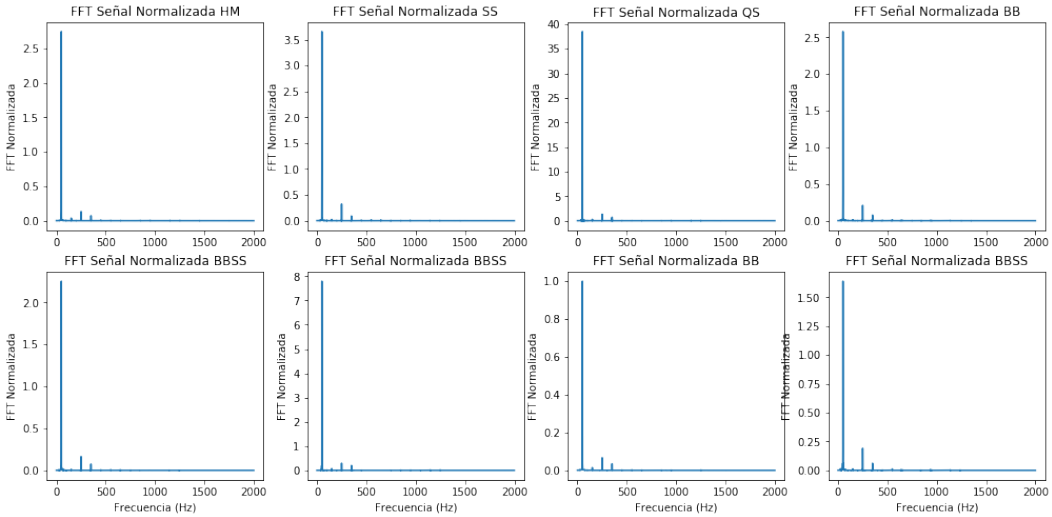


Figura 3.4: FFT de la señal normalizada

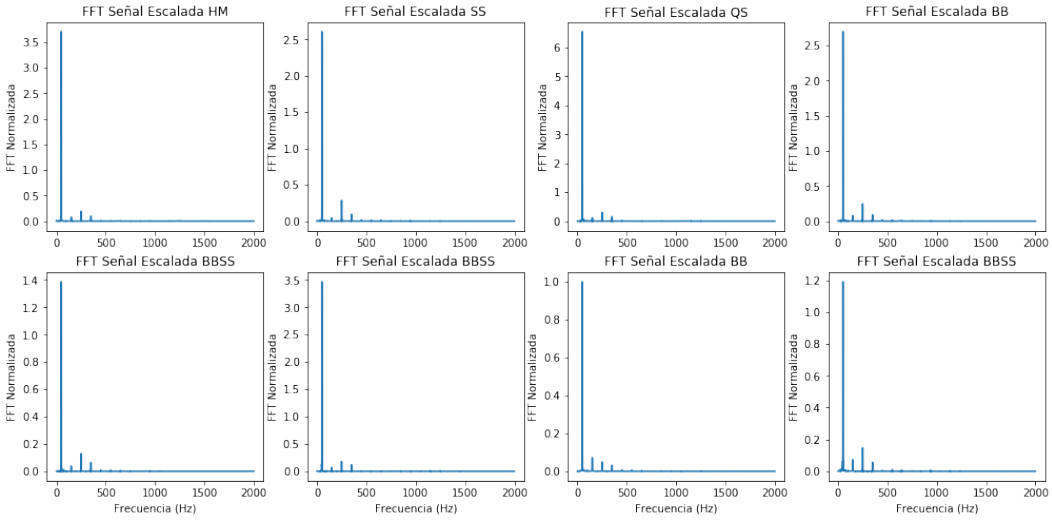


Figura 3.5: FFT de la señal escalada

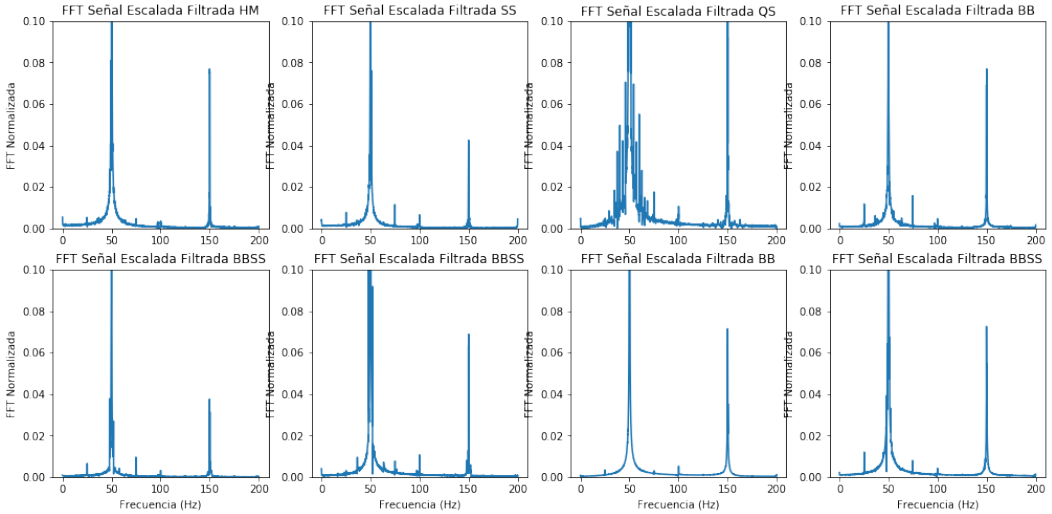


Figura 3.7: FFT filtrada de la señal escalada

De esta forma, se tiene las siguientes posibles entradas para los algoritmos de aprendizaje automático:

- La señal de la corriente de los motores de inducción: Sin modificar, normalizada y escalada.
- El valor absoluto de la señal de la corriente de los motores de inducción.
- El espectro de frecuencias de la señal; filtrada a 100 o 200 Hz.
- El espectro de frecuencias de la señal normalizada; filtrada a 100 o 200 Hz.
- El espectro de frecuencias de la señal escalada; filtrada a 100 o 200 Hz.

3.3. Metodología de los experimentos

En este apartado se explican las pruebas que se han realizado para detectar fallos en los motores de inducción, utilizando los distintos algoritmos de aprendizaje automático que se han planteado así como los datos preprocesados en el apartado anterior.

Primero, para comprobar la importancia que se le da a las amplitudes al momento de aplicar los algoritmos de aprendizaje automático, se hace una prueba rápida donde se utilizará KPCA al conjunto de muestras original, usando el valor absoluto de la señal como entrada del algoritmo. Se espera que el resultado de este algoritmo sea una separación de las muestras en clusters en función del nivel de carga del motor al momento de tomarse la muestra. Como se indicó en los apartados anteriores, al ser datos que se conocen a priori está separación no aportaría nueva información. Por lo tanto, en caso de cumplirse lo previsto, se descartará el valor absoluto de la señal como una entrada adecuada para el algoritmo.

Después de comprobar los resultados obtenidos usando el valor absoluto de la señal como entrada, se pondrán a prueba las 2 aproximaciones planteadas para poder detectar fallos en motores de inducción usando algoritmos de aprendizaje automático no supervisado. Estas aproximaciones (modelos) son:

1. Utilizar el algoritmo KPCA para ubicar las muestras en un nuevo espacio embebido y luego aplicar k-means para generar k clusters usando n componentes de KPCA. El kernel que se usará para KPCA será gaussiano y se especificará en cada prueba la cantidad de componentes principales que usará k-means para generar los clusters.
2. Utilizar el algoritmos SC para separar las muestras en k spectral clusters, las cuales se encuentran en el espacio embebido de los autovectores calculados por spectral clustering. Esta implementación de Spectral Clustering usará un grafo de similitud completamente conexo con kernel gaussiano y un laplaciano normalizado simétrico.

El objetivo de ambos modelos es obtener clusters que separen lo mejor posible las muestras en función del tipo de fallo que presenta el motor. Para esto, primero se tomará el conjunto de pruebas básicas (50 muestras y las clases equilibradas) y se probarán ambos modelos; en función de los resultados se pasará a probar con el conjunto de muestras original. La elección del conjunto de pruebas básicas es evitar que ante la baja cantidad de muestras de la clase de motores sin fallos (HM) en comparación con el resto de clases, los algoritmos tiendan a descartar estas muestras por enfocarse en las muestras de clases más prevalecientes en el conjunto de datos.

Por lo tanto, los experimentos seguirán los siguientes pasos:

1. Se seleccionará si los datos originales serán: normalizados, escalados o sin modificar.
2. Se indicará si se desea aplicar FFT a la señal y si se desea filtrar.

Percentil	Sigma	Gamma
10	0.736	0.923
25	1.296	0.298
50	2.045	0.120
75	2.508	0.079
100	3.129	0.051

Cuadro 3.5: Ejemplo de los valores de sigma y gamma calculados usando los percentiles de la matriz de distancias obtenida a partir del espectro de frecuencia obtenido de las señales normalizadas del conjunto de 100 muestras, filtrado a 100 Hz.

3. Se indicará si la señal o el espectro de frecuencias (aquello elegido en el paso 2) será normalizado, escalado o no se modificará.

4. Se usarán los datos del paso 3 como entrada de cada modelo. Así mismo, se indicarán los siguientes parámetros e hiperparámetros:

   - El número de clusters en los que se quieren agrupar los datos.
   - El número de componentes principales de KPCA que usará k-means para generar los clusters (solo para el modelo que usa KPCA).
   - El gamma (γ) a partir del cual se obtendrá el sigma (σ) del kernel gaussiano de KPCA y SC.

5. Se obtendrá un grupo gráficos que mostrarán la distribución de las muestras en el nuevo espacio embebido generado por su respectivo modelo. En estos gráficos se verá la distribución de cada tipo de muestra por cluster, indicando si todas las muestras de una misma clase han caído en el mismo cluster o se encuentran divididas en distintos clusters.

Así mismo, como se indica en el paso 4, cada algoritmo de aprendizaje automático requiere de distintos hiperparámetros para su correcto funcionamiento. Para ambos modelos se necesita elegir un gamma que necesita KPCA y Spectral Clustering para sus respectivos kernel gaussianos. Entre las opciones para la selección de gamma una buena opción es definir una rejilla en escala logarítmica y probar distintos valores. No obstante, otra aproximación para la selección del gamma es seleccionar un sigma a partir de los percentiles de la matriz de distnacias que se forma usando los datos de entrada del algoritmo, como se muestra en el Cuadro 3.5, y luego obtener el gamma a partir del sigma usando la formula $\gamma = \frac{1}{2\sigma^2}$. El uso de los percentiles como sigmas se ve justificado en que la matriz de distancias muestras a priori la relación de similitud entre las distintas muestras de motores antes de pasar por los algoritmos, y la anchura del kernel depende directamente de la distancia entre patrones. En la Figura 3.8 se muestra mediante un diagrama de calor los valores de la matriz de distancia, representando en color azul puntos muy cercanos y en tonos rojos puntos más lejanos. Se puede apreciar que las muestras que pertenecen a la misma clase definen bloques azules, mostrando que acorde al núcleo definido son más parecidos entre sí.

En resumen, los experimentos tendrán las siguientes etapas:

1. Se usarán todas las formas posibles de preprocesar las señales de los motores especificados en los apartados anteriores. Tal que, se puede tener: la señal original, la señal original normalizada, la señal original escalada y los respectivos espectros de frecuencia filtrados a 100 y 200 Hz.

2. Se lanzará una versión del experimento donde a los datos elegidos en la etapa 1

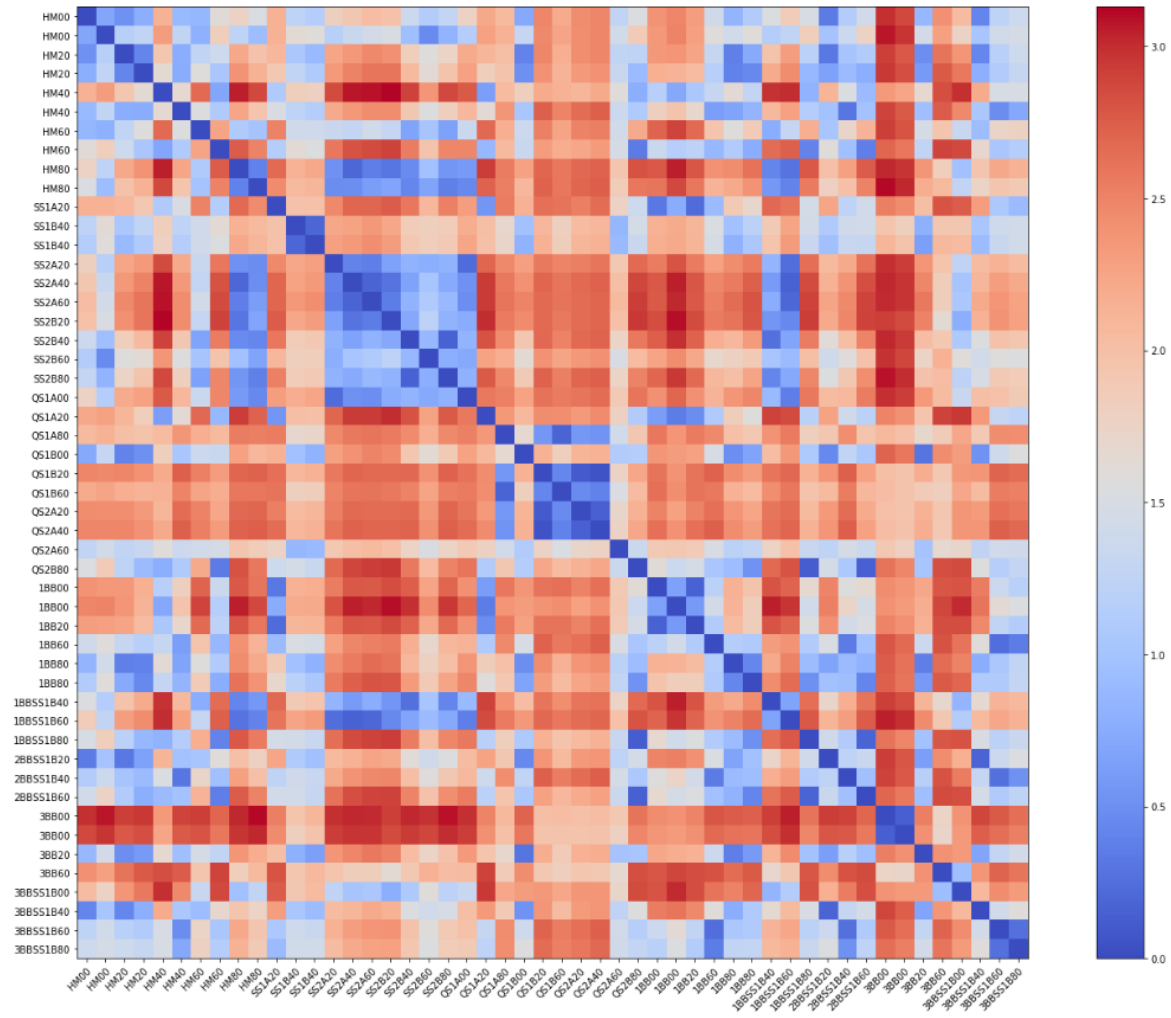


Figura 3.8: Heatmap que muestra la matriz de distancias correspondiente al espectro de frecuencia obtenido de las señales normalizadas del conjunto de 100 muestras, filtrado a 100 Hz.

se normalizarán, escalarán o no se modificarán antes de servir de entrada para los algoritmos.

3. Se calculará la matriz de distancias usando los datos de entrada obtenidos en la etapa 2 y se usaran los percentiles 10, 25, 50, 75 y 100 como sigmas a partir de los cuales se obtendrá un gamma. A estos gamma se les agregarán los múltiplos de 10 que van desde $1e^{-4}$ a $1e^{3}$. Una vez obtenido el listado de gammas final, se lanzará una versión del experimento por cada gamma en la lista.

4. Por último, antes de lanzar cada versión del experimento, se indicará al algoritmo que trate de separar las muestras en 4 y 5 clusters. Adicionalmente, solo para el modelo que usa KPCA y K-means, aparte de la cantidad de clusters también se indicará el número de componentes principales de KPCA que usará K-means para formar los cluster. Los valores a utilizar serán: 2, 3, 5 y 10 componentes.

5. Al terminar cada experimento se obtendrán un par de gráficos 2D que pinten las muestras en el nuevo espacio embebido, distinguiéndolas según el tipo de fallo y cluster donde se las ha ubicado; así como un listado con la cantidad de muestras de cada clase por cluster.

De esta forma se abarcan todas las posibles combinaciones de entradas para los modelos. De igual forma, el recalcular el gamma por cada nueva entrada solventa el caso de que si bien un determinado gamma da el mejor resultado para cierta entrada usando cierto modelo no significa que no hubiese otro gamma que diese un mejor resultado para el mismo modelo pero con otra entrada. Por eso mismo, también se varía la cantidad de clusters y el número de componente principales de KPCA que utiliza k-means para generar los cluster; ya que, por ejemplo, cierta versión del modelo puede separar mejor las muestras en 4 clusters usando 2 componentes mientras que otra versión lo puede hacer en 5 clusters usando 10 componentes. Dicho esto, para elegir los mejores resultados se evaluarán las gráficas obtenidas por cada versión del experimento, identificando si se han formado clusters que separen, en su mayoría, las muestras de distintas clases. Así mismo, una vez elegidos los mejores resultados obtenidos usando el conjunto de pruebas básicas (50 muestras) se probarán las mismas configuraciones con el conjunto de muestras original (151 muestras).

Por último, tanto los algoritmos de aprendizaje automático que se utilizan (KPCA, K-means y Spectral Clustering) como los métodos para escalar, normalizar y aplicar FFT a los datos corresponden a las implementaciones de los mismos por la librería Scikit-learn [11] en Python.

3.4. Resultados

En este apartado se muestran los resultados obtenidos en los experimentos estipulados en el apartado anterior.

En primer lugar, tal y como se esperaba, se obtuvo una correcta ubicación y separación de la muestras en clusters para el experimento que utilizaba el valor absoluto de la señal original de la corriente de los motores de inducción como entrada para KPCA, como se aprecia en la Figura 3.9. No obstante, como se previó que pasaría, esta separación se da en función del nivel de carga con el que funciona cada motor de inducción; y dado que ese es un dato conocido del motor, la información obtenida no es relevante.

En cuanto a los experimentos ejecutados usando el conjunto de pruebas básicas (50 muestras), el mejor resultado es el que se obtuvo usando el modelo de KPCA con los datos originales y aplicando k-means con 2 componentes principales para obtener 5 clusters. Este resultado se puede apreciar en la Figura 3.10, donde en la gráfica de la izquierda se ha coloreado los puntos según el tipo de fallo, en la del centro según el número de barras rotas y en la de la

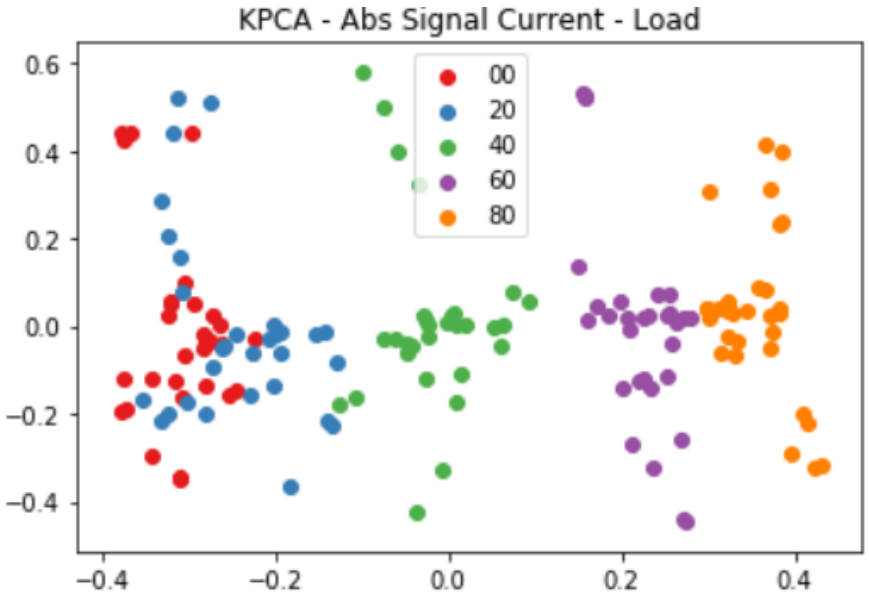


Figura 3.9: Salida de KPCA usando los valores absolutos de las señales de las corrientes

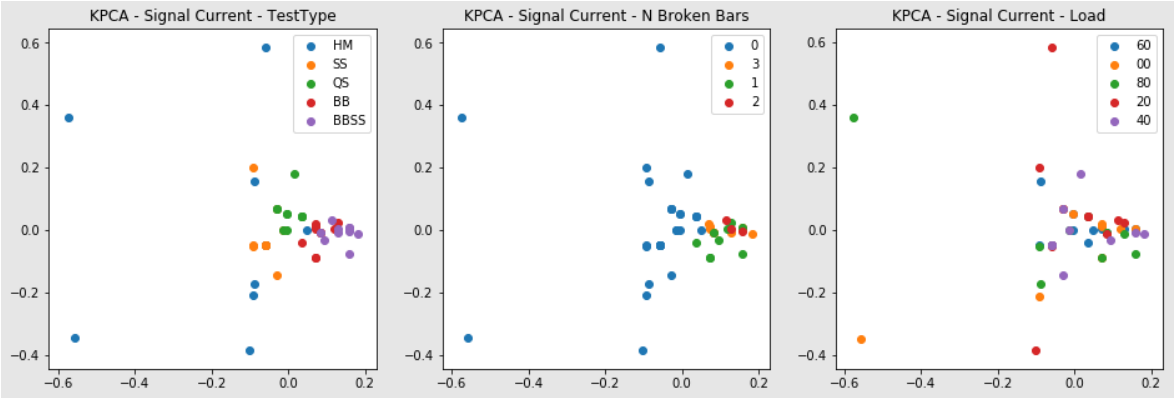


Figura 3.10: Mejor resultado obtenido por KPCA en el conjunto de pruebas básicas

derecha según el nivel de carga del motor; siendo en la primera gráfica donde claramente se distinguen clusters por cada tipo de fallo. Para analizar los clusters que ha generado k-means se revisa la Figura 3.11, donde lo más resaltante es el cluster 0, el cual contiene todas las muestras que corresponden a fallos por barras rotas (BB y BBSS). También se puede apreciar que todas las QS han caído en el cluster 0 y la mayoría de las SS han caído en el cluster 4. No obstante, si bien hay clusters como el 2 y 3 que están compuestos solo por HM, el resto de las muestras de los motores sin fallos se encuentran dispersos en los distintos cluster. Si bien en principio esto puede parecer algo negativo, al analizar con más detenimientos los clusters se aprecia que en los clusters 0 y 1 es solo 1 muestra de HM la que se cuela dentro del cluster. Esto último, sumado a las claras áreas de cada clase que se pueden identificar en la Figura 3.10, parecen indicar que en caso de disponerse de una mayor cantidad de muestras de HM mejoraría el clustering del algoritmo.

Por otro lado, los experimentos planteados que usan Spectral Clustering (SC) no han conseguido encontrar un modelo que genere un buen clustering como el que presenta el mejor caso de KPCA. Las Figuras 3.12, 3.13, 3.14 y 3.15 corresponden a los mejores casos obtenidos por el algoritmo SC para generar 5 clusters usando distintos preprocesamientos del conjunto de pruebas básicas como entradas. A diferencia de los clusters generados por

```
**************************** K-Mean: 5 clusters, 2 componentes, targetTested: TestType ****************************
```

Cluster	TestType	TestType value_counts
0	BB	10
	BBSS	10
	HM	1
1	QS	10
	SS	3
	HM	1
2	HM	1
3	HM	2
4	SS	7
	HM	5

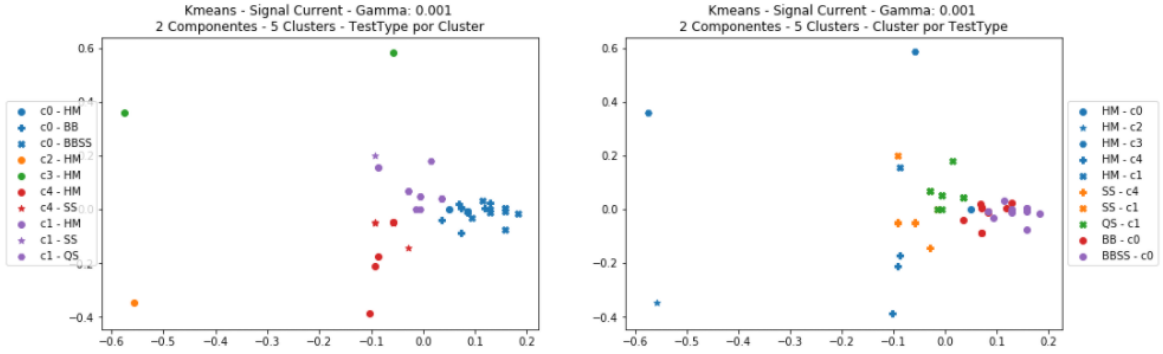
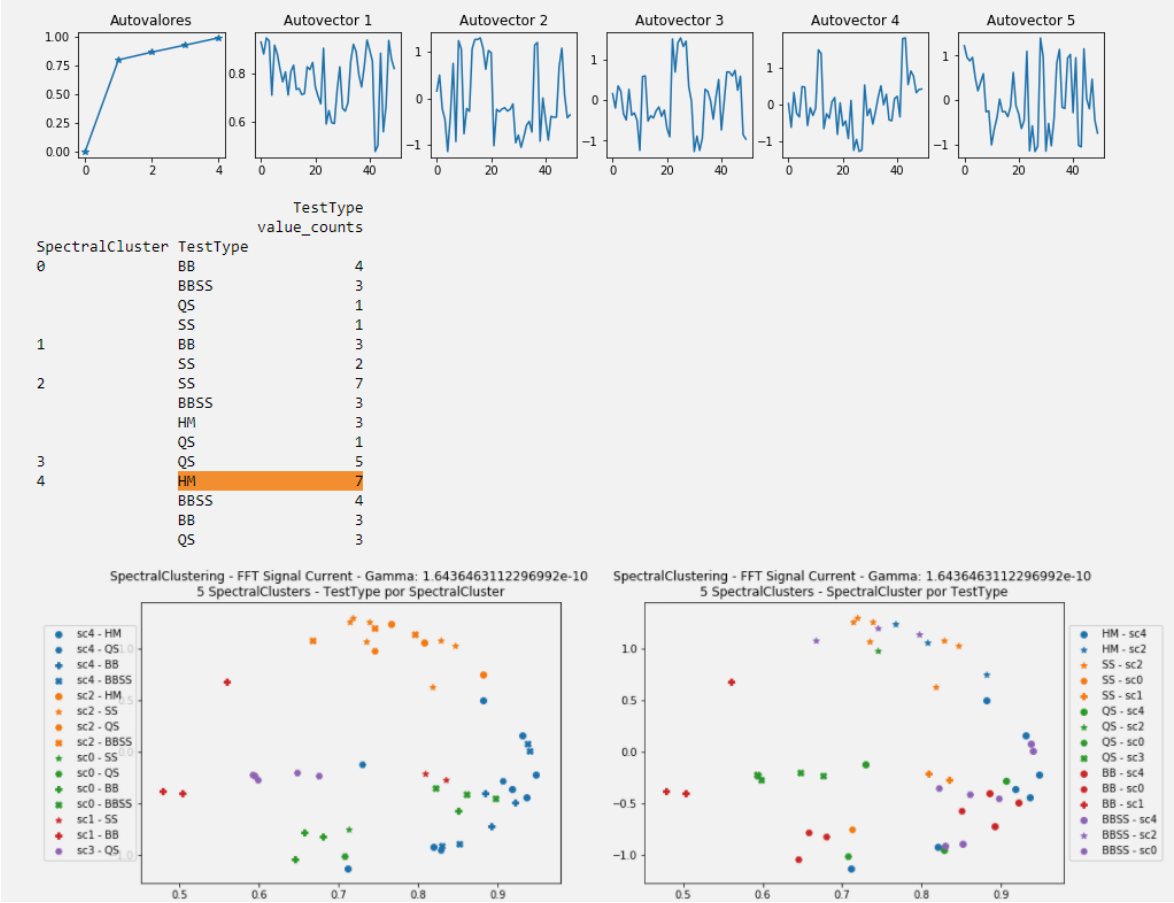


Figura 3.11: Clusters generados por kmeans a partir del mejor resultado obtenido por KPCA en el conjunto de pruebas básicas

el experimento de KPCA, en estos 4 resultados no se puede apreciar que haya un cluster claramente predominante para alguno de los tipos de fallos en particular. Respecto a las muestras de HM, si bien en cada uno de los 4 experimentos se puede apreciar que hay un cluster que contiene a la mayoría de las muestras sin fallos (7 de 10 HM); es importante resaltar que el mismo cluster siempre contiene de entre 3 a 4 muestras de otras tres clases distintas. Debido a esto se pasó a graficar los autovectores de cada uno de los 4 experimentos, para ver si se podía identificar algún autovector que diera más información sobre las diferencias entre las distintas clases; no obstante, salvo por algún pico en específico, estos no han proporcionado información nueva. No obstante, dado que la mayoría de los HM han caído en el mismo cluster, es posible que Spectral Clustering obtenga mejores resultados en caso de disponer de más muestras de HM.

Finalmente, se probó a utilizar el mejor modelo de KPCA encontrado usando el conjunto de pruebas básicas (50 muestras) para separar el conjunto de muestras original (151 muestras). Sin embargo, tras ejecutar el modelo no se obtuvieron los mismos resultados que obtuvo KPCA para el conjunto de pruebas básicas, como se puede ver en la Figura 3.16; no obteniendo ningún cluster que separase correctamente alguna de las clases del resto o separase los HM del resto de clases. El no obtener resultados similares entre los 2 conjuntos de muestras sustenta la suposición de que ante la baja cantidad de muestras de HM frente al resto de clases los algoritmos tienden a presentar un sesgo a favor de las otras clases, no consiguiendo una clara separación de las muestras de HM. Por otro lado, en el caso de Spectral Clustering, se decidió replicar los experimentos usando como entrada el conjunto de muestras original (151 muestras); no obstante, como se puede ver en la Figura 3.17 ,tampoco se encontraron resultados que satisficieran los objetivos del trabajo.



SpectralCluster	TestType	TestType value_counts
0	BB	4
	BBSS	3
	QS	1
	SS	1
1	BB	3
	SS	2
2	SS	7
	BBSS	3
	HM	3
	QS	1
3	QS	5
4	HM	7
	BBSS	4
	BB	3
	QS	3

Figura 3.12: Clusters generados por SC a partir del espectro de frecuencias de la señal normalizada filtrada a 100 Hz del conjunto de pruebas básicas

Figura 3.13: Clusters generados por SC a partir del espectro de frecuencias de la señal escalada filtrada a 100 Hz del conjunto de pruebas básicas

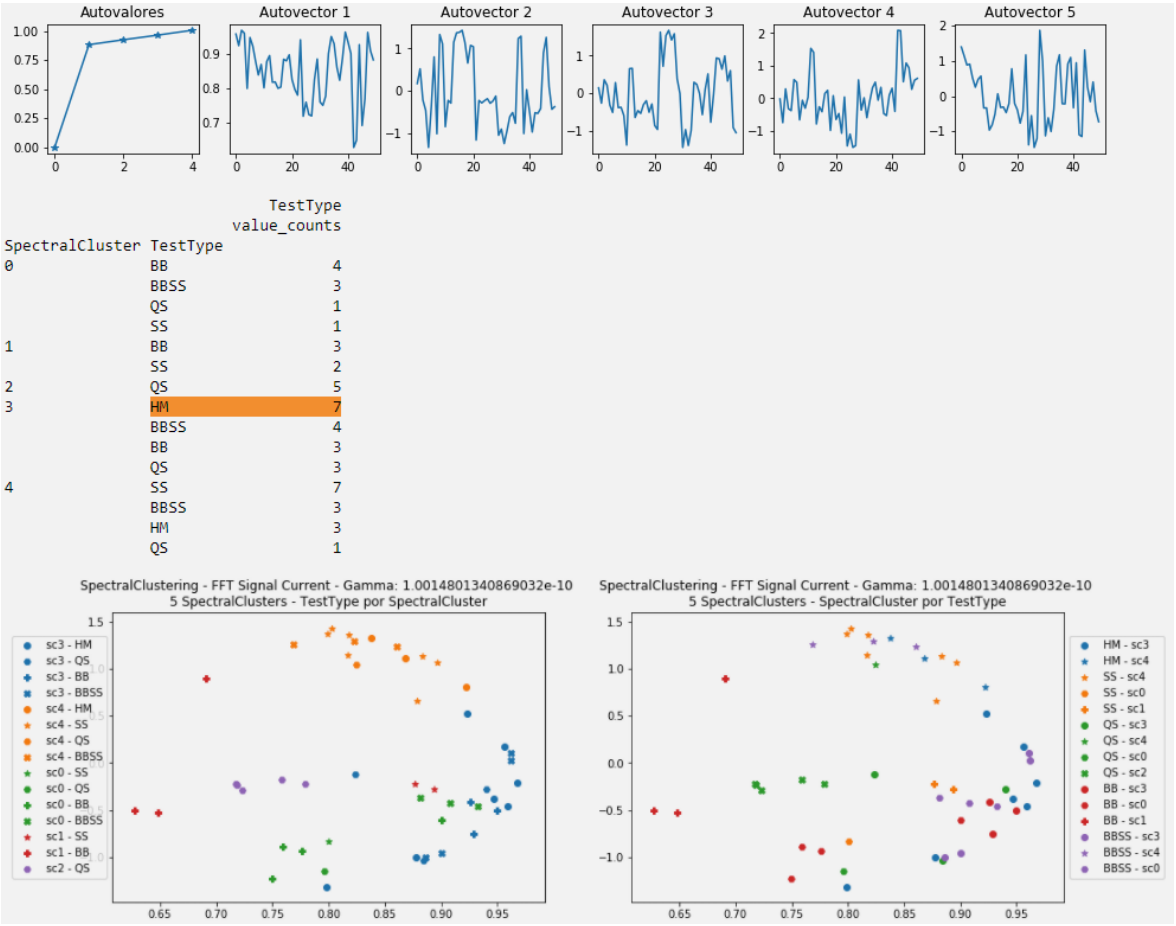


Figura 3.14: Clusters generados por SC a partir del espectro de frecuencias de la señal normalizada filtrada a 200 Hz del conjunto de pruebas básicas

Figura 3.15: Clusters generados por SC a partir del espectro de frecuencias de la señal escalada filtrada a 200 Hz del conjunto de pruebas básicas

```
**************************** K-Mean: 5 clusters, 2 componentes, targetTested: TestType ****************************

                   TestType
                value_counts
Cluster TestType
0       QS                9
        SS                6
        BB                4
        BBSS              4
        HM                1
1       SS                2
2       QS                1
3       QS                2
        BB                1
4       SS               32
        QS               28
        BBSS             27
        BB               25
        HM                9
```

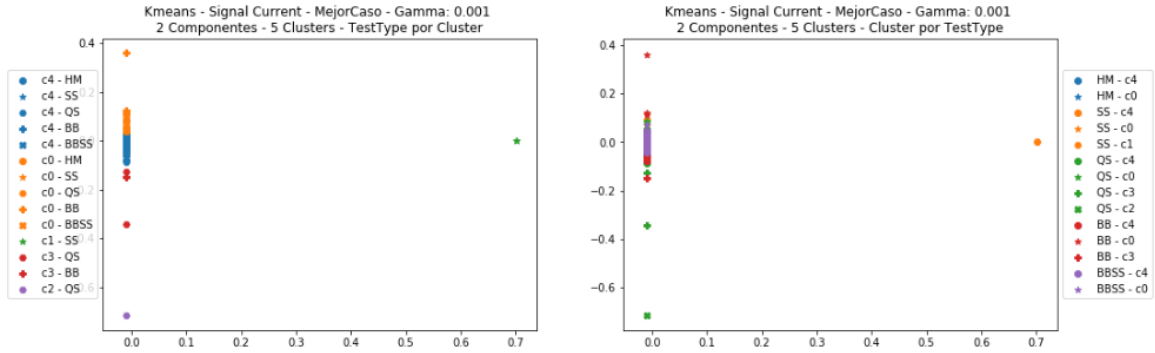


Figura 3.16: Clusters generados por kmeans a partir del mejor resultado obtenido por KPCA en el conjunto de muestras original

Figura 3.17: Clusters generados por SC a partir del espectro de frecuencias de la señal escalada filtrada a 200 Hz del conjunto de muestras original

Capítulo 4

Conclusiones y Trabajo Futuro

4.1. Conclusiones

En este trabajo se ha abordado el problema de la detección de fallos en motores de inducción mediante el uso de técnicas de aprendizaje automático no supervisado. Para ello se realizaron pruebas sobre un conjunto de muestras de la señal producida por la corriente de los motores de inducción; las cuales, luego de aplicárseles distintos preprocesos, se utilizaron como entrada de los algoritmos Kernel Principal Component Analysis (KPCA) y Spectral Clustering (SC). El objetivo de estos experimentos era conseguir un modelo que pueda separar, en distintos clusters, los motores sanos de lo motores con fallos. De igual forma, se busca que dicho modelo también pueda distinguir el tipo de fallo que sufre el motor.

De entre todos los experimentos ejecutados, el mejor caso que se obtuvo corresponde al modelo que utilizaba KPCA y la señal original de la corriente de los motores de inducción como entrada. Este modelo ha conseguido ubicar los datos en un espacio embebido tal que se puedan apreciar agrupaciones de los patrones de cada tipo de fallo, como se muestra en las Figura 3.10, y que a la vez se pudiesen separar en clusters donde hay una clara clase predominante por sobre las demás, como se muestra en la Figura 3.11 donde el cluster 0 contienen todos los motores que presentan fallos debidos a barras rotas. No obstante, no se han conseguido obtener los mismo resultados satisfactorios al replicarse el experimento usando este mismo modelo pero ahora con las señales originales del conjunto de muestras original (151 muestras) como entrada, obteniéndose los resultados de la Figura 3.11. Tras el análisis de los distintos experimentos, se cree que este hecho es probablemente debido al alto desbalanceo de las muestras y que se trata de un conjunto de datos pequeño.

Por otro lado, no se ha conseguido ningún modelo que utilice SC y que de unos resultados igual de satisfactorios que los indicados en el mejor caso de KPCA. En este caso los grupos obtenidos son más heterogéneos y, aunque se consiga ubicar en un mismo cluster a casi todas las muestras de HM, en este mismo agrupamiento aparecen mezcladas casi la mitad de las muestras de otras tres clases con fallos, como se puede ver en la Figura 3.15 para el conjunto de pruebas básicas y en la Figura 3.17 para el conjunto de muestras original.

Como se ha indicado previamente, un motivo por el que los algoritmos de aprendizaje automático no supervisado no han conseguido los resultados esperados puede ser el desbalanceo en la cantidad de muestras por cada clase en el conjunto de muestras original. Además, al tenerse solo 10 muestras de motores sin fallos (HM) frente a las 30 o 40 muestras en el resto de clases (motores con fallos), es posible que esto provoque que los algoritmos presenten un sesgo con respecto a las clases con mayor cantidad de muestras; dificultado la tarea de identificación y separación de las clases minoritarias, específicamente la clase

de los motores sanos. Por lo tanto, no se descarta la posibilidad de que ambos algoritmos de aprendizaje automático no supervisado obtuviesen buenos resultados al trabajar con un conjunto de muestras más grande y equilibrado.

Finalmente, dado que los resultados obtenidos por el modelo de KPCA usando como entrada las señales originales del conjunto de pruebas básicas (50 muestras) son lo suficientemente satisfactorios, se puede afirmar que es posible detectar fallos en los motores de inducción a partir de la señal de la corriente en lugar de usar el espectro de frecuencia.

4.2. Trabajo Futuro

A continuación, partiendo de las conclusiones obtenidas en este trabajo, se proponen las siguientes líneas de trabajo futuro:

- Se propone ampliar el conjunto de muestras de señales producidas por los motores de inducción. Al disponer de un conjunto de trabajo más amplio y equilibrado es probable que los algoritmos de aprendizaje automático no supervisado puedan desempeñarse mejor, evitando los sesgos propios del desequilibrio entre las clases. Especialmente se recomienda tener una mayor cantidad de muestras de motores sin fallos (HM) que del restos de clases que corresponden a motores con fallos.
- Se propone cambiar el enfoque de los experimentos y utilizar algoritmos de aprendizaje automático supervisados para tratar de clasificar los distintos tipos de fallos de motores. Este nuevo enfoque puede confirmar si las señales producidas por los distintos tipos de fallo en los motores de inducción son lo suficientemente distintas entre sí como para entrenar un modelo y realizar la clasificación de nuevas muestras. No obstante, este enfoque también requiere de un conjunto de datos mucho mayor para permitir un buen entrenamiento de estas técnicas.
- Se propone aplicar otras técnicas para el tratamiento de las señales producidas por los motores de inducción como:
  - Aplicar un filtro de ruido para eliminar las oscilaciones.
  - Aplicar un filtro en banda para quedarse con las frecuencias del espectro de frecuencias que varían según el tipo de fallo.
  - Calcular la relación señal ruido y añadirlo como una dimensión extra para los algoritmos.
  - Saturar el valor de la frecuencia predominante del espectro de frecuencias para que los algoritmos tengan en mayor consideración al resto de frecuencias.

Bibliografía

[1] C. M. Bishop, *Pattern recognition and machine learning*. Springer, 2006.

[2] T. Hastie, R. Tibshirani, and J. Friedman, *The elements of statistical learning: data mining, inference, and prediction*. Springer Science & Business Media, 2009.

[3] J. Bossio, C. De Angelo, and G. Bossio, "Self-organizing map approach for classification of mechanical and rotor faults on induction motors," *Neural Comput & Applic*, vol. 23, pp. 41–51, 2013.

[4] O. Sid, M. Menaa, S. Hamdani, O. Touhami, and R. Ibtiouen, "Self-organizing map approach for classification of electricals rotor faults in induction motors," in *2011 2nd International Conference on Electric Power and Energy Conversion Systems (EPECS)*, 2011, pp. 1–6.

[5] A. Bellini, F. Filippetti, C. Tassoni, and G.-A. Capolino, "Advances in diagnostic techniques for induction machines," *IEEE Transactions on Industrial Electronics*, vol. 55, no. 12, pp. 4109–4126, 2008.

[6] B. Schölkopf, A. Smola, and K.-R. Müller, "Kernel principal component analysis," in *Artificial Neural Networks – ICANN'97*, W. Gerstner, A. Germond, M. Hasler, and J.-D. Nicoud, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 1997, pp. 583–588.

[7] U. von Luxburg, "A tutorial on spectral clustering," *Statistics and Computing*, vol. 17, no. 4, pp. 395–416, Dec 2007.

[8] A. Y. Ng, M. I. Jordan, and Y. Weiss, "On spectral clustering: Analysis and an algorithm," in *Advances in Neural Information Processing Systems*. MIT Press, 2001, pp. 849–856.

[9] N. Cristianini, J. Shawe-Taylor, and J. Kandola, "Spectral kernel methods for clustering," *NIPS*, vol. 649-655, 01 2002.

[10] T. Kohonen, "The self-organizing map," *Proceedings of the IEEE*, vol. 78, no. 9, pp. 1464–1480, 1990.

[11] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.