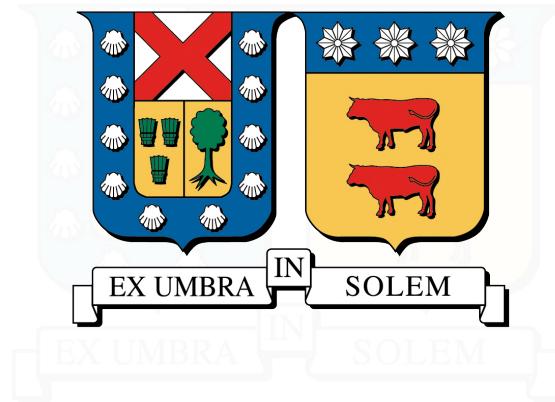


UNIVERSIDAD TÉCNICA FEDERICO SANTA MARÍA
DEPARTAMENTO DE INFORMÁTICA
SANTIAGO - CHILE



**EVALUACIÓN DE MODELOS DE APRENDIZAJE AUTOMÁTICO PARA
POSICIONAMIENTO INDOOR UTILIZANDO BLUETOOTH LOW ENERGY**

FELIPE IGNACIO BERRIOS TOLOZA

MEMORIA PARA OPTAR AL TÍTULO DE
INGENIERO CIVIL INFORMÁTICO

PROFESOR GUÍA : SR. HERNÁN ASTUDILLO ROJAS
PROFESOR CORREFERENTE : SR. HERNÁN BENAVENTE

MARZO 2018

RESUMEN EJECUTIVO

Lograr un posicionamiento exacto y con bajo error en interiores se ha convertido en una de las problemáticas más importantes del último tiempo, ya que otras tecnologías como GPS no se comportan de la forma esperada en este tipo de recintos, sobre todo en donde las señales les cuesta penetrar como edificios o lugares bajo tierra. Entonces, en esta memoria se presenta una forma de abarcar el problema del posicionamiento *indoor*, mediante las redes inalámbricas del protocolo Bluetooth con las cuales utilizando su intensidad de señal y en conjunto con algoritmos de máquinas de aprendizaje y la técnica *Fingerprint*, se plantea una solución para lograr posicionamiento en interiores. Para ello se realiza la experimentación en el estacionamiento subterráneo de la universidad Federico Santa María y se desarrolla una aplicación móvil Android para medir y probar los algoritmos. Los resultados muestran que el mejor algoritmo es redes neuronales con un error medio de 3.93 metros para el caso estático (sin movimiento) y k nearest neighbour con 6.68 metros en el caso dinámico (en movimiento). Además, se analizan las ventajas de usar técnicas de reducción de dimensionalidad como PCA, en donde la mayoría de los algoritmos se ven favorecidos, a excepción de redes neuronales. Finalmente se determina que redes neuronales es el mejor algoritmo, a pesar de su distribución dispersa, ya que presenta errores bajos y poco tiempo de procesamiento.

Palabras Clave. Posicionamiento Indoor, Maquinas de aprendizaje, Bluetooth Low Energy, Clasificación, Fingerprint.

ABSTRACT

Achieving accurate positioning and low error indoors has become one of the most important issues of recent times, since other technologies such as GPS do not behave in the way expected in this type of enclosures, especially where the signals It costs to penetrate as buildings or places underground. So, in this report we present a way of covering the problem of positioning indoor, by means of the wireless networks of the Bluetooth protocol with which using their signal intensity and in conjunction with algorithms of learning machines and the technique Fingerprint, a solution is proposed to achieve indoor positioning. For this, experimentation is carried out in the underground parking of Federico Santa María University and an Android mobile application is developed to measure and test the algorithms. The results show that the best algorithm is neural networks with an average error of 3.93 meters for the static case (no movement) and k nearest neighbor with 6.68 meters in the dynamic case (in movement). In addition, we analyze the advantages of using dimensionality reduction techniques such as PCA, where most algorithms are favored, except for neural networks. Finally, it is determined that neural networks is the best algorithm, despite its dispersed distribution, since it presents low errors and little processing time.

Keywords. Indoor Positioning, Machine Learning, Bluetooth Low Energy, Classification, Fingerprint.

Índice de Contenidos

1. Introducción	1
2. Definición del problema	3
2.1. Definición del problema	3
2.2. Objetivos	4
2.2.1. Objetivos Específicos	4
3. Estado del Arte	5
3.1. Tecnologías para posicionamiento <i>indoor</i>	5
3.1.1. Basado en Visión	5
3.1.2. Infrarrojo	6
3.1.3. Técnicas basadas en Sonido	7
3.1.4. <i>Wi-Fi</i>	8
3.1.5. <i>RFID</i>	9
3.1.6. <i>Bluetooth</i>	9
3.1.7. Otras tecnologías	9
3.1.8. Comparativa de tecnologías <i>Wireless</i>	11
3.2. Técnicas matemáticas <i>Wireless</i> para localización <i>indoor</i>	11
3.2.1. Proximidad	12
3.2.2. Triangulación	12
3.2.3. Fingerprint	14
3.2.3.1. Formulación del problema Fingerprint	16
3.2.3.2. Acercamientos convencionales de Fingerprint	17
3.2.3.3. Avances recientes en Fingerprint	18
3.3. Motivación de la selección de Fingerprint	20
4. Propuesta de solución	21
4.1. Consideraciones Previas	21
4.1.1. Definición de Beacons <i>Bluetooth</i> y sus protocolos	21
4.1.1.1. ¿Como se comunican los Beacons?	22
4.1.1.2. ¿Dónde comenzó la tecnología Beacon?	22
4.1.1.3. ¿En dónde pueden ser utilizados los Beacons?	23
4.1.1.4. Perfiles Beacon	23
4.1.2. Estabilidad de la señal <i>Bluetooth</i>	24
4.1.3. Evaluación y selección de Beacons <i>Bluetooth</i>	25
4.1.4. Algoritmos de <i>Machine Learning</i>	26
4.1.4.1. K-Nearest Neighbor	27
4.1.4.2. Support Vector Machines	29
4.1.4.3. Redes Neuronales y Deep learning	31
4.2. Descripción del <i>framework</i> de posicionamiento	33
4.2.1. Fase Offline	33
4.2.2. Fase Online	37

5. Experimentación	39
5.1. Beacons y configuración	39
5.2. Lugar de experimentación	42
5.3. Software Utilizado	42
5.3.1. Descripción de la aplicación	42
5.3.2. scikit-learn	43
5.3.3. Tensorflow	43
5.4. Recolección de Fingerprints	44
5.5. Visualización de los datos obtenidos	47
5.6. Análisis PCA y LDA	48
5.7. Entrenamiento de clasificadores	49
5.8. Entrenamiento utilizando PCA	54
5.9. Fase Online	59
6. Resultados	63
6.1. Métricas Obtenidas	63
6.2. Análisis de resultados	69
7. Conclusión	73
Bibliografía	77

Índice de Tablas

3.1. Comparativa Tecnologías Inalámbricas	11
4.1. Parámetros por defecto y métricas esperadas en los dispositivos Beacons Bluetooth	22
4.2. Tabla comparativa Estimote y Kontakt.IO	26
4.3. Estructura general del radiomap a construir	34
5.1. Tabla de resultados algoritmos de clasificación	53
5.2. Tabla resumen varianza explicada y acumulada PCA	56
5.3. Tabla de resultados algoritmos de clasificación PCA	59
6.1. Resultados método dinámico sin utilizar PCA	63
6.2. Resultados método dinámico utilizando PCA con 5 componentes	63
6.3. Resultados método estático sin utilizar PCA	64
6.4. Resultados método estático utilizando PCA con 5 componentes	64
6.5. Error con un CDF del 100 % para los algoritmos analizados en el método dinámico	66
6.6. Error con un CDF del 100 % para los algoritmos analizados en el método estático	67
6.7. Tabla Boxplot método dinámico	67
6.8. Tabla Boxplot método estático	68
6.9. Mejoras de tiempo en cada algoritmo	68



Índice de Figuras

3.1. Dispositivo volador 10 gramos	6
3.2. Proyección en murallas vision based	6
3.3. Método <i>Acoustic Background Spectrum</i>	8
3.4. Navegación mediante sensores inerciales <i>First Fit</i>	10
3.5. Comparativa tecnologías inalámbricas	11
3.6. Lateración y Angulación	12
3.7. Curvas Hiperbólicas TDoA	13
3.8. Proyección de vectores AoA	14
3.9. Proceso de creación Fingerprint	14
3.10. Robots para map Fingerprint	15
3.11. Fingerprint esquema básico	18
4.1. Tipos de paquetes Eddystone con su respectivo tamaño	24
4.2. Paquete iBeacon realizado por Apple	24
4.3. Estabilidad de la señal Bluetooth	25
4.4. Esquema básico machine learning	27
4.5. Clasificación utilizando K-NN	29
4.6. Support Vector Machines	30
4.7. Esquema básico de single layer perceptron	32
4.8. Redes neuronales profundas	33
4.9. Ejemplo grilla Fingerprint	34
4.10. Framework desarrollado para posicionamiento indoor	38
5.1. Beacons utilizados en la experimentación	39
5.2. Posicionamiento con diferentes intervalos de aviso	40
5.3. Listado de Beacons utilizados	41
5.4. Configuración Beacons para TX Power e Intervalo de transmisión	41
5.5. Plano del estacionamiento subterráneo	42
5.6. Lugar de experimentación real	42
5.7. Disposición de Beacons y puntos de referencia	45
5.8. Pestaña de configuración de la aplicación Android	46
5.9. Archivo CSV obtenido a partir de base de datos	46
5.10. Comparación de los diferentes algoritmos Manifold Learning	48
5.11. Comparativa entre los algoritmos PCA y LDA	49
5.12. Curvas de aprendizajes obtenidas por cada clasificador	51
5.13. Estructura de las redes neuronales construidas	52
5.14. Accuracy y Loss(Costo) obtenidos en el entrenamiento de la red neuronal profunda	53
5.15. Análisis de los valores propios obtenidos a partir de la matriz de covarianza	55
5.16. Proporción de la varianza explicada PCA	56
5.17. Comparativa de resultados obtenidos por los clasificadores utilizando PCA	56
5.18. Curvas de aprendizajes obtenidas por cada clasificador utilizando PCA	58

5.19. Accuracy y Loss(Costo) obtenidos en el entrenamiento de la red neuronal profunda con dos capas ocultas utilizando PCA	59
5.20. Pantallas utilizadas en la fase Online para la realización de las pruebas	60
5.21. Archivo obtenido en la experimentación online	61
6.1. Cummulative distribution function para el método dinámico	65
6.2. Cummulative distribution function para el método estático	66
6.3. Diagrama de cajas para método estático y dinámico	68



1 | Introducción

Al momento de definir la posición o localización, inmediatamente se manifiesta la idea de un sistema de referencia con sus respectivas coordenadas, es decir, un espacio de determinadas dimensiones, en el cual se puede medir y basado en un consenso común, establecer unidades para la correcta ubicación dentro de este.

La localización geográfica, es precisamente cualquier forma de localización dentro de un contexto geográfico. Desde la edad antigua, múltiples formas de localización han sido inventadas para ayudar a referenciar al ser humano, lo cual ha permitido facilitar así el comercio, la navegación y otros aspectos tan básicos y relevantes que hasta el día de hoy son requisitos para el correcto funcionamiento de la civilización.

Distintos tipos de indumentaria se utiliza para la navegación y localización, como son mapas, brújulas, relojes, telescopios, sextantes, entre otros. Dentro de los avances más importantes en este ámbito, es el desarrollo de la teoría científica y técnica denominada georreferenciación, la cual permite el posicionamiento espacial de una entidad en una localización geográfica y única, definida según un sistema de coordenadas y datum específicos. Tomando ventaja de lo anterior, es que se ha desarrollado el sistema de posicionamiento global o GPS, el cual mediante satélites y fusión de sensores puede georreferenciar cualquier dispositivo que presente un receptor GPS, sin importar el lugar del mundo en donde se encuentre y su conectividad. Gracias a GPS, el crecimiento y acceso de la georreferenciación y navegación está en progresivo aumento, a tal punto que cualquier persona con un smartphone puede saber su posición exacta, con un error de apenas unos pocos centímetros, con lo cual GPS es sin duda uno de los avances tecnológicos más grandes del último tiempo, ya que permite mejorar y optimizar rutas de comercio, flotas de transporte, ayuda además a ubicar direcciones, o también es fundamental en el uso militar o comercial.

Por todos los motivos anteriormente mencionado, GPS es la tecnología que lidera el posicionamiento en exteriores (*outdoor*), sin embargo carece de la posibilidad de georreferenciar cuando las condiciones son adversas, por ejemplo, cuando la entidad a localizar está bajo tierra, o dentro de un edificio de múltiples pisos, ya que las ondas emitidas por los satélites no son capaces de penetrar las estructuras y se desvanecen, lo cual no permite el posicionamiento en interiores (*indoor*).

La localización en interiores es sin duda uno de los problemas más desafiantes del último tiempo, esto es debido a múltiples inconvenientes que han sido detectados por los experimentadores, por lo que localizar en interiores no sea tan sencillo como lo es en exteriores, lo cual tiene a muchas empresas e investigadores desarrollando soluciones según diversas perspectivas para así lograr mejorar la exactitud del posicionamiento en interiores, y con ello, un estándar *indoor* como lo es GPS en exteriores.

La presente memoria abarca este tema, el problema del posicionamiento *indoor*, es decir, sin conectividad GPS en un lugar de experimentación en donde no se puede georreferenciar por los métodos comunes, lo cual establece la necesidad de buscar otras alternativas que presenten resultados favorables en este tipo de entornos y recintos.

La búsqueda de una solución en esta memoria nace a partir de la problemática de georreferenciar dentro de una explotación minera o mina, ya que, por el contexto, es difícil y costoso cablear para utilizar señales WiFi, por lo mismo, es necesario utilizar dispositivos más fáciles de instalar como son equipos Bluetooth y de bajo costo monetario. Además, dentro de la mina, las señales se ven sumamente afectadas por

el entorno y la interferencia, con lo cual se requieren técnicas matemáticas que sean capaces de aprender patrones y disminuir este ruido.

La presente memoria entonces abarca el tema de posicionamiento indoor utilizando Bluetooth, combinándolo a su vez con técnicas de máquinas de aprendizaje, para determinar qué tan efectivo es el posicionamiento, además de su exactitud y error. La estructura de este trabajo se describe a continuación: El capítulo 2 define más específicamente el problema en cuestión, tratando así el contexto y la motivación con algunos casos reales. Además, se definen los objetivos base y alcance de la memoria. Luego en el capítulo 3, se establece el estado del arte, en donde se explican en extenso las formas más conocidas de abarcar el problema de localización en interiores, las tecnologías empleadas y técnicas matemáticas desarrolladas, cada una con sus ventajas y desventajas a modo de comparativa. El capítulo 4 describe la propuesta de solución desarrollada a lo largo del texto, además presenta una pequeña introducción a los dispositivos Beacons Bluetooth, algoritmos de máquinas de aprendizaje y técnicas de reducción de la dimensionalidad. El capítulo 5 presenta la experimentación, es decir, el lugar de experimentación, software y hardware utilizado, formato y tipo de pruebas realizado y construcción de la aplicación para posicionamiento. El capítulo 6 presenta los resultados obtenidos con un respectivo análisis, para el estudio del mejor algoritmo de posicionamiento. Finalmente, el capítulo 7 presenta las conclusiones de la presente memoria, con algunas formas de extender y mejorar los actuales sistemas de posicionamiento a modo de recomendación para futuras investigaciones.

2 | Definición del problema

2.1. Definición del problema

En la actualidad, el sistema de posicionamiento global, GPS por sus siglas en inglés, se ha vuelto la tecnología de facto para el posicionamiento en exteriores, ya que permite una exactitud que puede alcanzar los 2 metros. El mayor problema radica en que GPS se basa en ondas con una frecuencia que habitualmente no traspasa objetos sólidos, con lo cual las señales enviadas por los múltiples satélites no traspasan algunas barreras con facilidad.

Cuando se usa tecnología GPS dentro de edificios o bajo tierra, existen muchos obstáculos e interferencia que empeoran significativamente la señal y exactitud, o simplemente las señales no pueden ser alcanzadas en el dispositivo. Con una frecuencia de 1575.42 MHz, GPS está calificado como una señal de alta frecuencia, por lo que su longitud de onda es corta y no puede atravesar objetos macizos.

Para realizar la aproximación de posición de un determinado dispositivo o terminal, GPS utiliza 24 satélites a unos 20.200 kilómetros de altura. Cuando un determinado dispositivo desea saber su ubicación, debe contar con al menos tres satélites en su rango de visión, y utilizando la trilateración, se obtiene una aproximación de unos pocos metros hasta inclusive unos pocos centímetros.

Entre las principales fuentes de error en la calidad de los datos de posición que afectan a GPS, se encuentran principalmente los errores propios del satélite, como por ejemplo su reloj interno, errores orbitales y errores de la configuración geométrica entre los satélites. Por otro lado, existen errores a la recepción del dispositivo, como el ruido asociado a los datos o el centro de fase de antena. Finalmente, y más relativo a este trabajo, están los llamados errores de propagación, que son asociados al medio en donde se transportan las ondas, afectando significativamente los resultados obtenidos.

Para solucionar este problema, existen los denominados *indoor positioning systems* (IPS), los cuales consisten en sistemas basados en redes inalámbricas, campos magnéticos, señales acústicas y otros métodos que utilizan los sensores internos de los teléfonos celulares. Algunos ejemplos de los problemas que resuelven estos sistemas son por ejemplo ayudar a encontrar tiendas dentro de un centro comercial, guiar personas discapacitadas visualmente en instalaciones o edificios, ayudar al transporte dentro de minas subterráneas para mover materiales. También se ha planteado sistemas de rescate en donde ante una eventualidad como por ejemplo un incendio, alguien alarma al sistema y este busca la salida más cercana dentro del edificio guiando al usuario (Rüppel et al., 2010).

Las tecnologías inalámbricas se han convertido en las más prometedoras dentro de los IPS, las cuales destacan debido al auge de los smartphones y la facilidad de utilizar redes como WiFi y Bluetooth. Muchas implementaciones de IPS se han realizado para WiFi, ya que este presenta un rango mucho más grande que Bluetooth, de hasta 35 metros, sin embargo, las señales en ocasiones pueden ser muy ruidosas. Bluetooth por su parte tiene un alcance en interiores de hasta 10 metros, pero con una precisión de hasta 1 metro. Además, la última versión de Bluetooth denominada *Bluetooth Low Energy* consume significativamente menos energía que WiFi.

Las técnicas actuales de posicionamiento *indoor* cuentan con problemas como la reflexión en múltiples objetos como paredes, muebles o el mismo cuerpo humano. Este problema se denomina “multi-path

propagation”, y es uno de los principales causantes de error en la localización en interiores. Los métodos que confían plenamente en el indicador RSSI, el cual mide la fuerza de la señal, están sujetos a errores inherentes a la variación de las señales y al ruido. Múltiples modelos matemáticos se utilizan para sobrellevar esta situación, entonces evaluar algoritmos que aprendan de la intensidad de las señales en cierta área es necesario para estimar de mejor manera la posición. Por lo tanto, el problema a abordar es mejorar la exactitud del posicionamiento *indoor*, para lo cual se utiliza técnicas de aprendizaje automático.

2.2. Objetivos

El principal objetivo de este trabajo es determinar los algoritmos de aprendizaje automático que presenten mejores resultados en términos de precisión y exactitud para posicionamiento *indoor* o en interiores, a partir de la experimentación en un recinto definido bajo ciertas condiciones experimentales.

2.2.1. Objetivos Específicos

- Evaluar calidad de las señales *Bluetooth Low Energy*, tanto en precisión como exactitud.
- Diseñar un método de mapeo para un área mediante señales RSSI (*fingerprint*).
- Comparar métodos de aprendizaje automático sobre mediciones RSSI para determinar cual posee menor error y es más exacto.
- Identificar límites del modelo propuesto, tanto es precisión, exactitud, error, distancia de alcance las señales, y otros factores variables.
- Determinar que tanto afectan los métodos de reducción de dimensionalidad tanto en precision, error y tiempo de procesamiento para los algoritmos de máquinas de aprendizaje estudiados.

3 | Estado del Arte

En los últimos años, el uso de sistemas de localización en interiores ha tomado mucha importancia en una gran cantidad de aplicaciones y contextos. En exteriores, la tecnología de facto es GPS, que sin embargo no presenta buenos resultados en interiores debido a la pobre o nula señal que es recibida en recintos cerrados o bajo tierra. A pesar de que la necesidad de posicionar o monitorear personas u objetos en interiores es alta, los avances no han sido significativos en este campo, debido a la gran cantidad de dificultades que se presentan al momento de resolver este problema como son por ejemplo propagación por múltiples caminos o *Non Line of Sight* (NLoS), es decir, sin línea de visión a los diferentes emisores de señales debido a las condiciones del sistema. También se presentan otros problemas por las condiciones cambiantes del ambiente, por ejemplo, múltiples dispositivos funcionando simultáneamente, alto tránsito de personas y objetos, lo cual atenúa en gran medida las señales o la dispersión de la radiación que es un fenómeno físico frecuente en las ondas de radio.

Claramente no existe una única solución que se adapte a todos los lugares o recintos, ya que cada uno de estos tiene características únicas o en ocasiones se requiere satisfacer ciertos parámetros como por ejemplo costos, exactitud, precisión, escalabilidad, entre otros. Por lo general no es posible optimizar simultáneamente todos estos parámetros nombrados anteriormente, por lo que habitualmente se evalúan las mejores combinaciones de ellos para obtener el mejor desempeño, adaptados a una solución personalizada dependiendo del ambiente en el cual se implementa el sistema de posicionamiento.

3.1. Tecnologías para posicionamiento *indoor*

Una gran cantidad de tecnologías han sido dispuestas para el desarrollo de soluciones al problema de posicionamiento en interiores. Las siguientes secciones muestran las más utilizadas y que a lo largo de múltiples estudios han presentado los mejores resultados.

3.1.1. Basado en Visión

Esta aproximación está basada en el uso de datos provistos mediante cámaras, en formato de vídeo o fotografías. Habitualmente se utilizan dos técnicas para su implementación:

Sistemas de cámara fijos: En este caso el ambiente es equipado con cámaras en diferentes puntos, procurando cubrir la totalidad del recinto. El objetivo entonces consiste en encontrar un objeto en movimiento capturado por una o varias cámaras. Las características del objeto a rastrear deben ser diferenciadas, con lo cual, cuando estas características sobresalientes están presentes en alguna de las cámaras, es posible obtener la distancia relativa, y por ende, al estar fijas las cámaras, la posición también es obtenida basado en la distribución espacial dentro de las imágenes capturadas.

Sistemas de cámaras en movimiento: El objetivo móvil está equipado con una cámara y la localización se realiza colocando varios puntos de referencia en posiciones conocidas (y orientaciones) o extrayendo características del entorno. Si la cámara móvil detecta dos o más puntos de referencia, puede descubrir su propia posición y orientación. Para ello, el proceso de localización implica dos etapas. En la etapa fuera de

línea, imágenes del ambiente se capturan en ubicaciones predefinidas y cada imagen se procesa para extraer sus características únicas para almacenarlas en una base de datos. En la etapa en línea, la cámara captura una imagen y se extraen sus características y se comparan con las características almacenadas para estimar la ubicación de la cámara.

Estas técnicas han alcanzado altos niveles de precisión, inferiores incluso a $10^{-6} [m]$ para sistemas de alta precisión ([Mautz y Tilch, 2011](#)). Por otra parte, la capacidad computacional de vídeo y las altas tasas de procesamiento de datos, sumado a los nuevos algoritmos de compresión y procesamiento de imágenes, hacen de esta tecnología una de las más prometedoras. Un punto desfavorable son sus altos costos, y poca escalabilidad, sin embargo, nuevas formas de utilizar las cámaras de dispositivos móviles están siendo investigadas para desarrollar soluciones de bajo costo manteniendo la eficiencia.

Uno de los acercamientos más osados en el posicionamiento en interiores, es la localización mediante información visual, es decir, basado en imágenes relativas al lugar de exploración. Para el desarrollo de estos modelos habitualmente se usa equipos de vuelo muy pequeños y livianos, los cuales recorren la zona tomando fotografías con el objetivo de *mapear* el recinto. En ([c. Zufferey et al., 2006](#)), los autores utilizan robots de aproximadamente 10 gramos y con una velocidad de vuelo de 1.5 m/s. Estos robots tienen todo lo necesario para regular su velocidad y prevenir colisiones. Estos micro robots están equipados con giroscopios y dos pequeñas cámaras, un pequeño radio control y un módulo Bluetooth. La Figura 3.1 muestra como están constituidos estos pequeños dispositivos voladores.

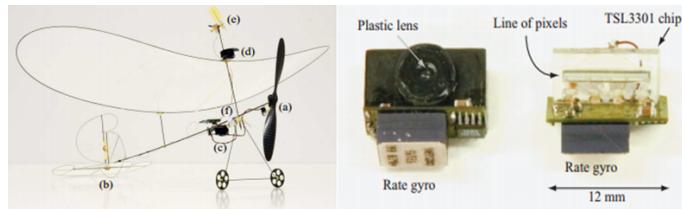


Figura 3.1: Dispositivo volador que toma imágenes de las murallas para detectar los patrones de textura

(Fuente: ([c. Zufferey et al., 2006](#)))

Estos dispositivos capaces de sobrevolar un recinto pequeño o casas determinan según las texturas de las paredes y su distorsión, la posición relativa dentro del recinto con respecto a la pared fotografiada y utilizando el giroscopio, previenen choques. En la Figura 3.2 se observa cómo se generan los patrones de texturas en las murallas para realizar las pruebas.

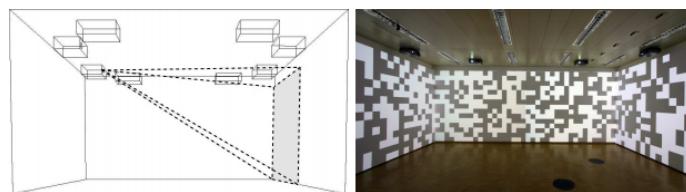


Figura 3.2: Proyectores que generan los patrones en las murallas con los cuales los autores realizaron las pruebas

(Fuente: ([c. Zufferey et al., 2006](#)))

Los resultados muestran que la solución es factible, sin embargo no puede controlarse la altitud, y para ello los autores plantean el desarrollo de un *3D visión based tracking system*.

3.1.2. Infrarrojo

La tecnología Infrarroja es una de las más utilizadas para localizar personas y objetos mediante el uso de emisores y receptores infrarrojos.

Una forma posible de utilizar la radiación infrarroja en sistemas de posicionamiento *indoor* es poner en el objeto o persona a localizar, un transmisor infrarrojo con un identificador único. Luego, los receptores son colocados en lugares dentro del recinto, los cuales pueden detectar este identificador único y comunicar a un software especializado, el cual se encarga de calcular la posición mediante la distancia entre el transmisor y receptor. Esta tecnología inalámbrica presenta grandes ventajas. En primer lugar, al no poder atravesar paredes, es fácil confinar las ondas en un recinto. Por otra parte, la radiación infrarroja no se ve afectada por la interferencia electromagnética. A pesar de lo anterior, la tecnología infrarroja presenta dos puntos que lo tornan una opción menos factible. Primero, se ve muy afectado por el efecto de *multipath-propagation*. Además, su implementación requiere hardware y software costoso, por lo que no es tan accesible.

3.1.3. Técnicas basadas en Sonido

La tecnología de ultrasonido utiliza las ondas de ultrasonido para medir la distancia entre una estación base que se encuentra fija, y el objeto a localizar. Para implementar este tipo de sistema, múltiples receptores deben estar sincronizados. Esta sincronización se logra mediante el uso de ondas de radio o radiación infrarroja, ya que estas son mucho más rápidas que las de ultrasonido. El funcionamiento consiste en que los transmisores envían ondas de radio y de ultrasonido al mismo tiempo. Las ondas de radio alcanzan casi instantáneamente a los receptores, con lo cual se pueden sincronizar. Posteriormente cada receptor detecta las ondas de ultrasonido al momento en que arriban, y con esta diferencia de tiempo, es posible calcular la distancia a cada receptor, así se genera una posición relativa dentro de la sala de medición.

Las ventajas de este sistema son los bajos costos de implementación, y la capacidad de funcionar a pesar de la mayoría de las obstrucciones presentes. Las desventajas por su parte son la alta complejidad de desarrollar esta tecnología en sistemas de gran escala, por la sincronización de los receptores. Además la temperatura es uno de los factores más críticos en la velocidad del sonido, es por ello que la gran mayoría de los sistemas basados en ultrasonidos incluyen sensores para compensar estas fluctuaciones de temperatura. Otros factores menos influyentes incluyen la presión del aire, su contenido de dióxido de carbono y la amplitud del sonido.

Otra forma de utilizar el espectro de sonido, y el ultrasonido, es lo que se denomina *acoustic background spectrum*.

En este caso, los autores sugieren una metodología la cual no requiere infraestructura [Tarzia et al. \(2011\)](#), es decir las redes inalámbricas son prescindibles. El método consiste en un mapeo *fingerprint* o huella, la cual registra el sonido ambiente de un determinado lugar, y luego construye una base de datos de todas las mediciones realizadas. Posteriormente para estimar la posición del usuario, realiza una comparativa con su actual *fingerprint* y lo contrasta con la base de datos, eligiendo de esta manera la posición más “cercana” según el algoritmo de estimación seleccionado.

El objetivo principal es determinar la posición mediante un dispositivo móvil, como puede ser un Smartphone, de manera rápida y económica a una resolución de una sala de mediano tamaño. La principal ventaja es que no se requiere prácticamente infraestructura por detrás, solo basta los sensores de sonido del equipo móvil.

El principal problema a resolver es entonces traducir los *fingerprint* a *roomlabels*, es decir la medición actual registrada por el dispositivo móvil, mapearla a una etiqueta de la posición dentro de la sala. Para la realización del *fingerprint* los autores determinan que este debe ser:

- Distintivo
- Responsivo
- Compacto
- Eficientemente computable
- Robusto al ruido
- Invariante al tiempo

Para ello crean *Acoustic background Spectrum* (ABS) que cumple estas condiciones ya que tiene en cuenta ruidos típicos de la vida cotidiana como puede ser sonidos de computadoras, aire acondicionado, entre otros. Para la obtención del *fingerprint* elaboran diversas técnicas basada en el estudio de los sonidos como se aprecia en la Figura 3.3.

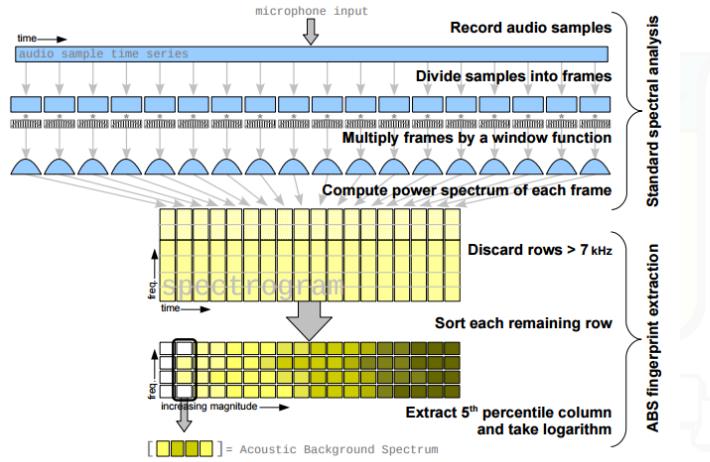


Figura 3.3: Descripción del método ABS con el cual se realiza la obtención del mapeo *fingerprint*

(Fuente: Tarzia et al. (2011))

Luego de obtener el *fingerprint* los autores usan algoritmos de clasificación, es decir aprendizaje supervisado, como es nearest-neighbor. Se realizaron pruebas en 33 habitaciones distintas, obteniendo una exactitud de 69 %, lo cual muestra resultados aceptables para el posicionamiento utilizando sonidos. Finalmente concluyen que dos *fingerprint* menos exactos pueden ser mejorados si se usan en conjunto, como es el caso de ABS con WiFi.

3.1.4. Wi-Fi

Wireless Local Area Network (WLAN) puede usarse para estimar la ubicación de un dispositivo móvil dentro de un recinto, dado que la infraestructura WLAN está muy extendida de acuerdo con el aumento de la demanda de redes de comunicaciones, por lo que este enfoque es ampliamente utilizado para posicionamiento *indoor*. Por esta razón, una de las principales ventajas del uso de la técnica de localización Wi-Fi es su rentabilidad ya que no se requiere adicionar infraestructura en lugares donde ya se encuentran estas redes. Otra ventaja de usar WLAN es que no se requiere una línea de visión con los emisores de señales. Es el método más popular que utiliza *Received Signal Strength Indicator* (RSSI), o intensidad de la señal recibida dentro de las tecnologías inalámbricas y que en este caso son extraídas de las redes IEEE 802.11

En una WLAN, un nodo emite / recibe señales de radiofrecuencia hacia / desde el enrutador inalámbrico, que se puede usar para determinar ubicación precisa de cualquier dispositivo con Wi-Fi habilitado utilizando técnicas que son detalladas posteriormente en esta memoria.

Es la tecnología inalámbrica más utilizada en el mundo, además por seguir un estándar, está claramente definida y sus protocolos funcionan en cualquier parte del mundo. Sus frecuencias son 2.4 GHz y 5 GHz, además es muy escalable utilizando access point, y está muy masificado en dispositivos móviles. Por lo anterior, Wi-Fi se ha vuelto la alternativa más utilizada en técnicas de posicionamiento en interiores.

Recientes investigaciones demuestran que es posible alcanzar una precisión de aproximadamente 3-5 metros (Chen et al., 2014). Uno de los problemas más grandes son la necesidad de cablear la infraestructura para los routers y el alto consumo de energía que se presenta en ellos. Otro problema presente en la mayoría de las tecnologías inalámbricas es la atenuación de la señal por murallas, objetos o el mismo cuerpo humano.

3.1.5. *RFID*

La tecnología RFID se basa en lectores RFID con una o más antenas, y tags activos o pasivos. Los tags activos poseen una batería, con lo cual son capaces de emitir señales autónomamente, mientras los tag pasivos no poseen batería y requieren de incentivos externos para emitir sus señales. Típicamente se puede poner datos dentro de cada tag, los cuales ayudan a determinar las posiciones. Los datos almacenados dependen estrictamente de la memoria de los tags respectivos.

La localización mediante RFID puede categorizarse en dos tipos, los cuales son localización del lector y localización de tags. En la localización del lector RFID, su precisión depende estrictamente de la densidad de tags dispuestos en el recinto, es decir, la cantidad sobre el área cubierta. Este método funciona de la siguiente manera, el usuario porta un lector y lee los tags dispuestos en distintas posiciones. Cada tag posee información de la posición relativa, con lo cual es fácil reconocer la posición del usuario. Una desventaja obvia, es la poca escalabilidad debido al gran número de tags RFID.

El segundo método, consiste en determinar la localización del tag RFID. En este caso el usuario no porta un lector, sino que un tag, y los lectores son puestos en lugares estratégicos dentro del recinto, así cuando el usuario transita, los lectores son capaces de determinar en qué lugar fue leído el tag asociado al usuario y determinar su posición. Este método es incluso más costoso que el anterior, ya que los lectores RFID son más caros que los tags.

Las restricciones más grandes de la tecnología RFID son el corto alcance que este presenta y la posibilidad de leer solo unos pocos tags a la vez. Sin embargo, sus ventajas son muchas, entre ellas se encuentran la alta seguridad de transmisión, alta tasa de transferencia, no requiere línea de visión directa y presenta relativamente bajos costos.

3.1.6. *Bluetooth*

Bluetooth se ha convertido en el estándar para redes inalámbricas de área personal, WPAN por sus siglas en inglés. Opera en la banda de 2.4 GHz, sin embargo, posee menor alcance que otras tecnologías de radio frecuencia como por ejemplo WLAN, aun a pesar de ello se han registrado implementaciones a gran escala de posicionamiento *indoor* utilizando esta tecnología ([Nashville GOB, 2017](#)). Su rango típicamente alcanza entre 10 a 15 metros, ya que es pensado principalmente para comunicaciones directas entre emisor y receptor dentro de un rango corto, como puede ser una casa o una sala de mediano tamaño. A pesar de esto, el creciente interés de múltiples aplicaciones ha vuelto a la tecnología Bluetooth muy relevante, ya que hoy en día está presente en smartphones, computadores, notebooks entre otros. Con esto presente, no se requiere adicionar mayor hardware a los teléfonos celulares para implementar un sistema de localización, ya que prácticamente todos estos dispositivos cuentan con la tecnología Bluetooth integrada.

Bluetooth es una tecnología de bajo consumo energético y bajo costo, por lo que es eficiente y factible en el desarrollo de sistemas de localización *indoor*. Los tags bluetooth son pequeños y por esta razón pueden ser colocados en múltiples lugares para asegurar la cobertura de una región. Los tags poseen un ID único como puede ser la MAC u otro identificador generado por el fabricante.

En los últimos años, nuevos dispositivos han sido creados con el objetivo de ayudar al crecimiento del Internet de las cosas. Estos dispositivos son conocidos como Bluetooth beacons y muchas empresas han creado su propio hardware, además proveen kits de desarrollo de software para hacer más fácil la integración de esta tecnología en sistemas. Empresas como Apple, Kontakt.io, Estimote, entre otras, venden estos tags sin ninguna restricción en cuanto a su uso, por lo que existen múltiples casos de uso como se puede ver en ([Kontakt, 2017](#)).

3.1.7. Otras tecnologías

Existen muchas otras tecnologías, que en este momento son menos relevantes pero que sin embargo son investigadas frecuentemente para crear nuevos sistemas de posicionamiento o para mejorar y complementar

los existentes. Entre las más relevantes se puede destacar ZigBee, *Ultra Wide Band* (UWB), GSM, radio FM y sensores iniciales de los dispositivos móviles.

La localización en interiores utilizando **ZigBee** se basa en la creación de una red de sensores con posiciones conocidas, y un sensor ZigBee denominado *target* del cual se desea conocer su posición. Ya que estos nodos sensores pueden comunicarse entre sí, la fuerza de la señal recibida por los sensores es comúnmente usada para localizar al *target*. Se han utilizado múltiples algoritmos de localización con resultados favorables dependiendo del ambiente como se puede observar en ([Fang et al., 2012](#)).

UWB usa un pulso de radio de un sub-nanosegundo para transmitir datos en un amplio ancho de banda. Por estas características, se considera que no interfiere en otras señales de radio, por lo tanto, puede utilizarse en cualquier espectro de emisión. Utiliza bajo consumo de energía, y además en teoría no es afectado por *multi-path propagation*, es decir, la interferencia que se produce en las señales al rebotar o reflectarse en ciertas superficies.

Las ondas de radio **FM** son utilizadas ya que pueden eventualmente evitar algunas complicaciones presentes en otras tecnologías inalámbricas como Wi-Fi utilizando los mismos algoritmos.

GSM es utilizado principalmente en redes celulares, y cubre diferentes tipos de frecuencias, además está en prácticamente todos los edificios y ciudades, con lo cual no requiere mayor infraestructura. A diferencia de FM tiene menor distancia de propagación, por lo cual si sirve en interiores. El principal problema es que GSM está fuertemente patentado y es complejo realizar ensayos y pruebas sobre él.

Finalmente, los sensores iniciales se refieren a sensores que ocupan la inercia para medir aceleración como puede ser giroscopio, acelerómetro. En este contexto, los sistemas de posicionamiento basado en sensores son sistemas que explotan los sensores de movimiento y rotación, para continuamente calcular la posición, velocidad y orientación sin la necesidad de utilizar hardware o referencias externas.

La gran parte de los equipos móviles actuales cuenta con acelerómetro y brújula, con lo cual se puede realizar una navegación estimando ciertos parámetros. En ([Link et al., 2011; Tarrío et al., 2011](#)) , los autores plantean un modelo basado en los sensores de los dispositivos, ya que al ser capaces de detectar los pasos, se puede establecer una ruta predefinida dentro de un determinado lugar, pero la única desventaja es que es el usuario quien debe decirle al dispositivo la posición inicial o algún punto de referencia.

Los autores generan sus mapas con OpenStreetMap, luego realizan un algoritmo llamado *First Fit*, para detectar la dirección del movimiento y realizar el *matching* con la ruta predefinida, con ello se puede determinar la posición relativa del usuario al lugar en donde comenzó. La Figura 3.4 muestra esta situación.

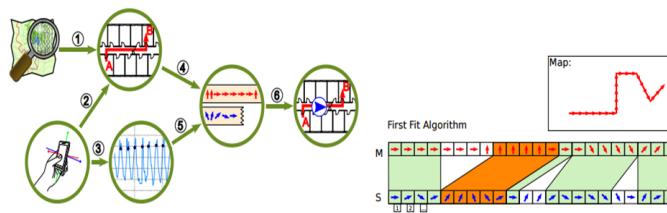


Figura 3.4: (a) A la izquierda, proceso completo para la navegación utilizando acelerómetro y conteo de pasos. (b) A la derecha, método First-Fit para anticipar y predecir posición del usuario.

(Fuente: ([Link et al., 2011; Tarrío et al., 2011](#)))

First Fit realiza una especie de *lookahead* para establecer el match con los pasos de la ruta y si los pasos que está detectando actualmente son errores. Con esto, puede mirar los pasos futuros y aproximar en donde se encuentra el usuario.

Los autores evalúan su propuesta en tres distintos escenarios, primero en exteriores para medir exactitud, luego en interiores para establecer si la solución es robusta, y luego en un escenario relativamente grande pero que es *indoor*. Los autores concluyen que la solución es factible para la navegación, pero no para el posicionamiento, ya que es el usuario quien ingresa su posición estimada según ciertos nodos dentro del

edificio. El sistema es escalable, pero requiere realizar las rutas predefinidas, lo cual puede ser mucho trabajo en edificaciones demasiado grandes.

3.1.8. Comparativa de tecnologías Wireless

Dado que los mejores resultados presentes en la literatura corresponden a las tecnologías *Wireless*, ya sean de sonido, radiofrecuencia, entre otros, a continuación, se muestran las ventajas y desventajas de cada una de ellas.

Tabla 3.1: Tabla comparativa tecnologías inalámbricas

	Precisión [m]	Rango [m]	Costo	Complejidad	Ambiente
Infrarrojo	10^{-2} - 1	1-5	Alto	Baja	Indoor
Ultrasonido	10^{-2}	2-10	Medio	Baja	Indoor
Wi-Fi	1-10	20-50	Medio	Baja	Indoor/Outdoor
RFID	10^{-1} -1	1-10	Bajo	Baja	Indoor
Bluetooth	1-10	1-30	Bajo	Baja	Indoor/Outdoor

Ninguna tecnología presentada anteriormente es completamente efectiva en todos los escenarios, por lo que la elección de estas se debe hacer según las características del lugar, como su dinámica de cambio, obstáculos presentes, tamaño, entre otros. En la Figura 3.5 se resume el alcance y características de cada tecnología de posicionamiento.

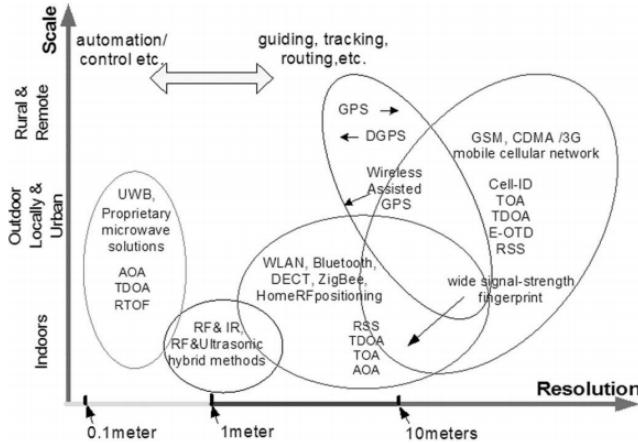


Figura 3.5: Comparativa de distintas tecnologías inalámbricas basado en su resolución y alcance.

(Fuente: ([Liu et al., 2007](#)))

3.2. Técnicas matemáticas Wireless para localización indoor

No es sencillo determinar un modelo completo para la propagación de ondas en interiores debido a muchos problemas que se presentan como lo es multipath, baja probabilidad de la línea de visión (LOS), y otros factores determinantes como tipo de material del edificio, superficies que reflejan ondas, el mismo cuerpo humano y objetos en movimiento. Por lo mismo es necesario establecer modelos matemáticos que reduzcan el error generado por la influencia de los factores antes mencionados.

Habitualmente se categorizan las técnicas matemáticas para la localización en interiores en 3 grandes grupos: Proximidad, Triangulación y *fingerprint* (*scene analysis*).

3.2.1. Proximidad

Es el método más simple, y se basa en determinar una posición simbólica y aproximada de la posición del usuario. Para lograrlo, existe una grilla de antenas o emisores de ondas de radio, y según la señal más fuerte detectada por el usuario, es donde se localiza en el sistema. Este tipo de localización es ampliamente usado en redes celulares, ya que permite determinar la posición de un dispositivo con una precisión de 50-200 m, sin embargo, no es buena en espacios reducidos. Por lo mismo se dice que este método tiene una alta varianza, y muchas veces no satisface la necesidad de localización. La ventaja es que al ser simple puede utilizarse en la mayoría de las redes inalámbricas, como lo es infrarrojo, RFID, Cell-ID, GSM.

Por lo anterior, habitualmente se dice que esta técnica es para saber en qué espacio geográfico o región está el usuario, no para saber su posición exacta (Liu et al., 2007).

3.2.2. Triangulación

La triangulación utiliza las propiedades geométricas de un triángulo para estimar la posición del objeto. La triangulación se divide en 2 puntos: lateración y angulación. Por una parte, la lateración estima la posición midiendo las distancias hacia múltiples puntos de referencia también llamados beacons. Angulación por su parte, no utiliza las distancias, sino que los ángulos relativos a cada beacon o punto de referencia. La Figura 3.6 muestra cómo se calculan las posiciones relativas con lateración y angulación.

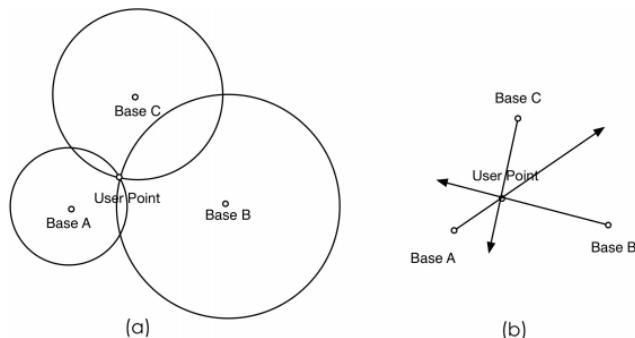


Figura 3.6: Técnicas de triangulación basadas en lateración (a) y angulación (b)

(Fuente: (Liu et al., 2007))

Si se conoce las *base station* emisoras de ondas, lateración puede calcular la posición basado en un modelo matemático en donde se utiliza la intersección de tres círculos. Por otra parte, angulación utiliza la información del ángulo de llegada de las señales y con esto puede trazar 3 vectores, en donde la intersección representa la posición del objeto receptor.

El principal problema es establecer la distancia para lateración, y los ángulos para angulación. Para ello se han diseñado muchos acercamientos que permiten obtener esos valores.

1. Técnicas de lateración

- a) **ToA (Time of arrival):** En ToA, la distancia del objeto móvil hacia las estaciones es directamente proporcional al tiempo de propagación, y además se sabe la velocidad de propagación de las ondas, que habitualmente es la velocidad de la luz siempre y cuando se transmitan en el aire. El dispositivo del usuario envía paquetes con las marcas temporales, es decir la hora en que se envía el paquete, luego la estación beacon puede determinar la hora en que llega el paquete, y haciendo

la diferencia se obtiene el tiempo de propagación. ToA presenta dos grandes problemas, uno es que todos los dispositivos deben estar altamente sincronizados, y segundo, es que los paquetes deben estar etiquetados con información de tiempo de envío ([Fang, 1990](#)).

- b) **TDoA (Time difference of arrive):** Es similar a ToA, sin embargo en este caso no se necesita el tiempo de emisión del transmisor, solo se requiere el tiempo de llegada a cada uno de las estaciones base o beacons y mediante la diferencia de tiempo entre ellas establecer la posición. Para lograrlo, cada TDoA, se representa como una curva hiperbólica de las posibles posiciones asumiendo distintos tiempos de transmisión como se muestra en la Figura 3.7 . Donde se intersectan las curvas de a lo menos 2 estaciones beacons, es donde se transmitió la señal, por lo mismo es la posición del usuario.

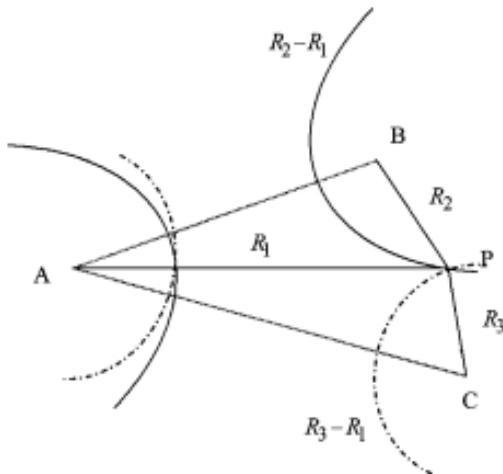


Figura 3.7: Curvas hiperbólicas de TDoA que estiman la posición basado en el tiempo de transmisión.

(Fuente: ([Liu et al., 2007](#)))

Habitualmente por la complejidad del método, se utilizan linealización de las fórmulas del hiperboloide, a través de expansión en series de Taylor para luego crear algoritmos iterativos computacionalmente eficientes como se muestra en ([Torrieri, 1984](#)).

- c) **Método basado en RSS (Received signal strength):** Los métodos anteriores tienen algunos inconvenientes, primero, es complejo encontrar una línea de visión entre el emisor y receptor, además de la propagación por ondas de radio que puede presentar efectos de multicaminos, es decir, una misma onda llega al receptor por dos vías distintas, debido a la reflexión u otros factores. Una alternativa es estimar la distancia utilizando la atenuación de la fuerza la señal.

Existen modelos teóricos y empíricos, los cuales intentar predecir la distancia utilizando la diferencia entre la fuerza de la señal entre el transmisor y el receptor. Este método no es muy utilizado, debido al desvanecimiento de las señales en interiores, sin embargo puede utilizarse usando algunos pre procesamientos o mejoras, como por ejemplo establecer contornos predefinidos de señales RSS en forma de elipses como muestran los autores en ([Zhou et al., 2005](#)).

2. Técnicas de angulación

- a) **AoA (Angle of arrival):** Como se define en ([Veen y Buckley, 1988](#)) , la localización del objetivo puede realizarse mediante la intersección de múltiples líneas con una dirección dada por el ángulo entre cada estación y el receptor. Los métodos AoA deben utilizar al menos dos puntos de referencia o estaciones beacon, y los dos ángulos respectivos con respecto al objeto que se quiere realizar la localización. La principal ventaja es que requiere tantas estaciones como el número de dimensiones en donde se requiere localizar el objeto, por ejemplo si se necesita

determinar la posición en tres dimensiones, se requieren tres estaciones bases. Además, no se necesita sincronización de tiempos.

Por otra parte, es complejo y difícil de implementar, además de requerir software especializado, ya que para una localización muy precisa el ángulo debe ser medido con muy poco error y esto es difícil de lograr. Al igual que otras técnicas se ve afectado por propagación *multipath* y desvanecimiento de las señales, interferencia y por la dirección de la apertura de medición. La Figura 3.8 muestra el comportamiento de AoA.

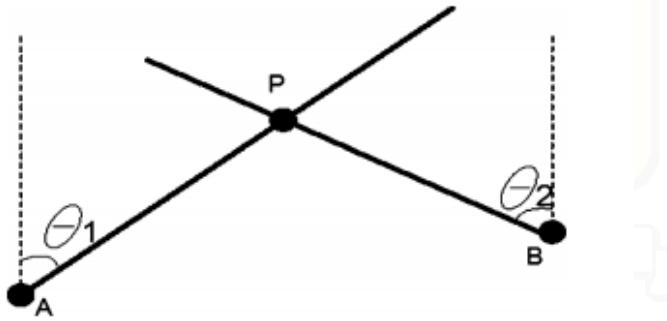


Figura 3.8: Proyección de vectores basado en ángulo de incidencia de las ondas sobre el objeto o dispositivo móvil.

(Fuente: ([Liu et al., 2007](#)))

3.2.3. Fingerprint

Se denomina *fingerprint* o *scene analysis* al tipo de algoritmos que en primer lugar recolecta características de un lugar y luego estima la localización en tiempo real utilizando las características, contrastándolas con las características actuales. Lo más habitual es utilizar Fingerprint basado en RSS.

Fingerprint presenta dos etapas, la etapa offline y la etapa online. Durante la etapa offline se realiza un reconocimiento de las características del lugar, en donde se mide las coordenadas de cada estación base y la fuerza de la señal en determinados lugares, obteniendo así un mapa de señales del entorno. Durante la etapa online se utiliza la señal actual recibida de cada estación base y un modelo generado en la etapa offline, generando así una estimación de la posición del usuario. La Figura 3.9 muestra el funcionamiento a nivel general de Fingerprint:

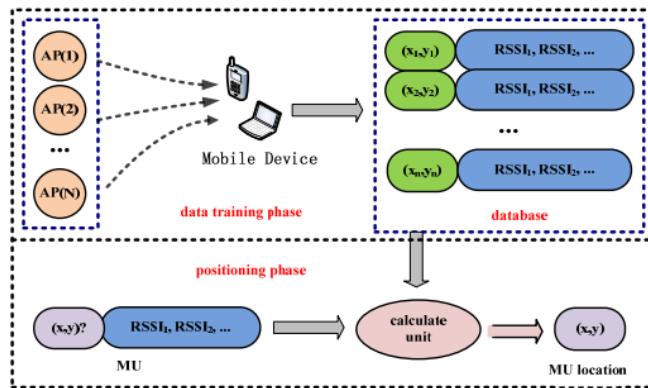


Figura 3.9: Proceso de captura de características Fingerprint y elaboración del modelo para estimar la posición.

(Fuente: ([Liu et al., 2007](#)))

Fingerprint se divide en dos grandes aristas, *map-based Fingerprint* y *map-free Fingerprint*:

1. Map-based Fingerprint

Este caso utiliza la etapa offline y online, en donde en la etapa offline se generan los radio-map de cada posición, es decir, asociar una característica de las señales según la posición geográfica en donde se toma la medición. Existen dos tipos de radio-map, los cuales son radio-map de tipo valor medio, y radio-map de tipo función de densidad de probabilidad. En el radio-map de tipo valor medio existe un número limitado de puntos en donde se toman las mediciones de las señales, mientras que el radio-map de tipo función de densidad de probabilidad, puede representar todos los puntos del lugar como una función continua. Para la estimación de la posición se han realizado variados acercamientos como es modelos de probabilidad, kNN, redes neuronales, SVM, estimación de máxima verosimilitud, estimación bayesiana ([Liu et al., 2007](#); [Ni et al., 2003](#)).

Para la generación de los radio-maps se utiliza habitualmente tres técnicas, las cuales son *Model Based Estimation*, Medición Offline y calibración Online. *Model Based Estimation* no confía plenamente en las señales RSS, si no que utiliza distintos modelos de propagación como son *Wall attenuation factor* ([Bahl y Padmanabhan, 2000a](#)) , 2D-ray tracing y otros modelos de propagación de las señales para estimar la fuerza de la señal en determinados puntos del espacio y generar el radio-map. La siguiente ecuación describe la fórmula de la atenuación de señales por el efecto de las paredes.

$$P(d)[dBm] = P(d_0)[dBm] - 10n\log\left(\frac{d}{d_0}\right) - \begin{cases} nW * WAF & \text{si } nW < C \\ C * WAF & \text{si } nW \geq C \end{cases} \quad (3.1)$$

En donde n representa la velocidad con la que aumenta la pérdida de poder de la señal con la distancia. $P(d_0)$ es el poder de la señal en cierto punto de referencia d_0 y d es la distancia que separa al transmisor y receptor. C es el máximo número de paredes para el cual el factor de atenuación hace alguna diferencia. nW es el número de paredes y WAF es el factor de atenuación por las paredes. Habitualmente n y WAF se conocen, ya que depende del lugar y orden de los objetos dentro de un edificio.

Por otra parte, se tiene mediciones offline, en donde se realiza un proceso lento y tedioso, de medir cada punto del lugar en donde se realiza la localización. Mientras más puntos se registran, mejor será el radio-map ya que tiene un mayor número de datos de entrenamiento. Para reducir el trabajo, algunos autores han utilizado robots para registrar la medición de las señales ([Yeh et al., 2009](#)). La Figura 3.10 muestra cómo se utiliza un robot para tomar mediciones de las señales en determinados puntos, en este caso los beacons corresponden a tags RFID.

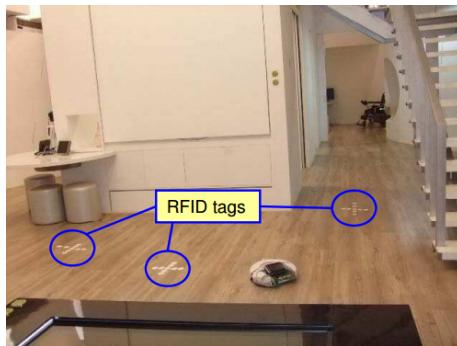


Figura 3.10: Robot que toma automáticamente las características para elaborar el mapeo fingerprint de un determinado lugar.

(Fuente: ([Liu et al., 2007](#)))

Por último, calibración online tiene en cuenta que se requiere mucho esfuerzo para realizar las mediciones y obtener las características de señales, además si el entorno cambia constantemente es probable que las señales también cambien y todo el registro se pierda y deba ser realizado nuevamente.

Entonces calibración online combina las dos técnicas anteriores, dejando ciertos nodos ancla que miden constantemente la señal en esos puntos, y mediante *model based estimation*, pueden estimar la fuerza de la señal en otros lugares en donde no hay nodos ancla. Los nodos ancla pueden realizar mediciones cada cualquier intervalo de tiempo, ajustando el modelo, y por lo mismo no se ve afectado por cambios en el entorno.

2. Map-Free Fingerprint

Map-based fingerprint puede utilizarse con cualquier red inalámbrica o incluso otros tipos de Fingerprint, como ultrasonido, campos magnéticos, entre otros. Sin embargo la complejidad para registrar mediciones aumenta cuando el área a mapear aumenta, además la mantención temporal del radio-map es un proceso engorroso y lento. Map-free reduce esta complejidad, utilizando otras formas de localizar usando fingerprints.

En Wang et al. (2012) el autor observa patrones de las señales en ciertos lugares como el ascensor o el pasillo, y con estos puntos de referencia utiliza aprendizaje no supervisado para estimar la posición del usuario. Combinando esta información junto con los acelerómetros y sensores del dispositivo móvil, el autor muestra cómo puede obtener una posición estimada muy cercana a la real. Entonces este tipo de Fingerprint, se basa en el patrón del punto de referencia actual y las mediciones de puntos de referencia anteriores. Por lo mismo se dice que este método no soporta localización estática.

3.2.3.1. Formulación del problema Fingerprint

En la técnica de *Fingerprint* el área es dividida en un conjunto de puntos de referencia (RPs), $P = \{p_j = (x_j, y_j) | j = 1, \dots, N\}$ en donde P define el conjunto cartesiano de RPs, que no necesariamente son un conjunto de puntos equidistantes entre si (Khalajmehrabadi et al., 2017). El dispositivo móvil graba un conjunto de mediciones *Fingerprint RSS* en los instantes t_m , $m = 1, \dots, M$ con magnitudes RSS $r_j^i(t_1), \dots, r_j^i(t_M)$ en cada RP, donde i indica el índice del access point desde el conjunto de access points $\mathcal{L} = \{AP^1, \dots, AP^L\}$. Generalmente se toma el mismo número de ejemplos de entrenamiento, M en cada RP. Los Fingerprints RSS de todos los access point en el tiempo t_m en el punto p_j están organizados en un vector $r_j(t_m) = [r_j^1(t_m), \dots, r_j^L(t_m)]^T$. El radio map completo guardado en el instante t_m puede ser reconocido por la siguiente matriz:

$$R(t_m) = (\mathbf{r}_1(t_m), \dots, \mathbf{r}_N(t_m)) = \begin{bmatrix} r_1^1(t_m) & r_2^1(t_m) & \dots & r_N^1(t_m) \\ r_1^2(t_m) & r_2^2(t_m) & \dots & r_N^2(t_m) \\ r_1^3(t_m) & r_2^3(t_m) & \dots & r_N^3(t_m) \\ r_1^4(t_m) & r_2^4(t_m) & \dots & r_N^4(t_m) \end{bmatrix}, \quad m = 1, \dots, M. \quad (3.2)$$

Un subconjunto de RPs con el mayor grado de similaridad es denotado como \mathcal{K} y está definido como $|\mathcal{K}| = K$. Esta similaridad es definida por cada método y es detallada posteriormente.

En la fase online el usuario recibe una medición RSS $\mathbf{y} = (y^1, \dots, y^M)^T$. Entonces el objetivo es simple; encontrar la localización $\hat{\mathbf{p}} = (\hat{x}, \hat{y})$, basado en una regla que compara la medición online contra el radio map de *fingerprints* como:

$$\hat{\mathbf{p}} = f(\mathbf{R}, \mathbf{y}) \quad (3.3)$$

En donde R denota la colección de radiomaps en todas las instancias guardadas. A continuación, se muestran los tres acercamientos principales para la localización utilizando Fingerprint.

3.2.3.2. Aceramientos convencionales de Fingerprint

1. **Aceramiento determinista:** En esta forma de resolver el problema de Fingerprint, la posición que logra la mayor similaridad con la posición real es alcanzada utilizando la siguiente formulación:

$$\hat{\mathbf{p}} = \operatorname{argmin}_{j=1,\dots,N} d(\check{\mathbf{r}}_j, \mathbf{y}) \quad (3.4)$$

En donde $\check{\mathbf{r}}$ es el valor representativo del Fingerprint en el j -esimo RP (Bahl y Padmanabhan, 2000b) y $d(\check{\mathbf{r}}_j, \mathbf{y})$ define una medida de distancia típica (Farshad et al., 2013). En el caso de promedio de tiempo, el valor representativo es el promedio de los Fingerprints durante el tiempo en que fueron medidos. La distancia Euclíadiana también puede ser utilizada, definida como:

$$d(\check{\mathbf{r}}_j, \mathbf{y}) = \|\mathbf{y} - \check{\mathbf{r}}_j\|_2 \quad j = 1, \dots, N. \quad (3.5)$$

Ya que el número de access point disponibles varía en la superficie a localizar, la posición es típicamente estimada sobre los AP visibles y los AP desconocidos se estiman en un valor que indica una señal débil o un valor imposible de obtener (100 dBm).

2. **Aceramiento probabilista:** En este acercamiento, una sola medida RSS puede ser no suficiente para estimar la posición, ya que no representa la variación temporal de los datos en la propagación en interiores. El desempeño del acercamiento determinista puede ser mejorado si en vez de utilizar solo un subconjunto de Fingerprints RSS, todos los Fingerprints son utilizados. En esta presunción se basa el acercamiento estadístico, utilizar todos los Fingerprints para describir de mejor manera el área en cuestión.

La forma en que el acercamiento estadístico estima la posición, se basa en la estimación de Máximo a Posteriori (MAP) (Honkavirta et al., 2009), de esta manera MAP estima la posición del usuario maximizando la probabilidad condicional de la posición dado la medición recibida en la etapa online:

$$\hat{\mathbf{p}} = \operatorname{argmax}_{j=1,\dots,N} f(\mathbf{p}_j | \mathbf{y}) \quad (3.6)$$

En donde $f(\mathbf{p}_j | \mathbf{y})$ es la probabilidad condicional de que el usuario este en \mathbf{p}_j dado que recibió el vector \mathbf{y} en la etapa online.

Gracias al teorema de Bayes esto puede ser re formulado de la siguiente manera:

$$f(\mathbf{p}_j | \mathbf{y}) = \frac{f(\mathbf{p}_j, \mathbf{y})}{f(\mathbf{y})} = \frac{f(\mathbf{y} | \mathbf{p}_j) f(\mathbf{p}_j)}{\sum_{j=1}^N f(\mathbf{y} | \mathbf{p}_j) f(\mathbf{p}_j)} \quad j = 1, \dots, N \quad (3.7)$$

La probabilidad $f(\mathbf{p}_j)$ es la distribución de la posición del usuario sobre el área completa, y se asume generalmente uniforme ya que no existe conocimiento previo en la posición del usuario, por lo que todos los puntos estudiados son equiprobables. Por lo anterior, $f(\mathbf{p}_j)$ puede ser ignorado en el problema de maximización, de esta forma el denominador en la ecuación 3.7 es el mismo para todo $j = 1, \dots, N$ de esta manera, el problema MAP es transformado a :

$$\hat{\mathbf{p}} = \operatorname{argmax}_{j=1,\dots,N} f(\mathbf{y} | \mathbf{p}_j) \quad (3.8)$$

Conocido comúnmente como el estimador de máxima verosimilitud o *Maximum Likelihood (ML)*.

Lo anterior revela que el acercamiento determinista confía en estimar la densidad a priori $f(\mathbf{y} | \mathbf{p}_j)$. Existen dos formas utilizadas en la estimación de esta distribución de densidad de probabilidad, las cuales son denominadas paramétrica y no paramétrica. La estimación paramétrica trata a la distribución

de probabilidad como una distribución analítica conocida, por ejemplo una distribución Gaussiana ([Haeberlen et al., 2004](#)) para aproximar las características temporales de las señales RSS. Este enfoque ha sido cuestionado en múltiples investigaciones, debido a lo poco realista que se torna este supuesto en muchos lugares y recintos experimentales. Los primeros acercamientos utilizaban una aproximación log-normal para la distribución, sin embargo, es sesgada y estacionaria sobre pequeños intervalos de tiempo. Las aproximaciones modernas de estimación paramétrica utilizan funciones Kernel para lograr mejores resultados.

La estimación no paramétrica no asume nada con respecto a la distribución de Fingerprint RSS. En vez de esto, la distribución es generada utilizando un histograma que cuadra con el radio map construido ([Haeberlen et al., 2004; Ladd et al., 2002](#)). En este emparejamiento, todos los datos son cuantizados en múltiples niveles y la frecuencia de cada barra del histograma es calculada, para la estimación de $f(\mathbf{y} | \mathbf{p}_j)$. El histograma consiste entonces en la concatenación de todas estas barras. Sin embargo, un gran número de ejemplos son necesarios en diferentes instantes de tiempo en cada RP para generar el histograma, lo que es complejo de obtener en la práctica.

3. **Técnicas de reconocimiento de patrones:** La idea básica de reconocimiento de patrones se basa en clasificadores, que son utilizados con los Fingerprints recolectados y luego son capaces de discriminar una nueva medición RSS desconocida durante la fase online. En la fase de entrenamiento de cada clasificador, el sistema ajusta los parámetros del modelo, mediante el uso del radio map de Fingerprints recolectados en forma de base de datos. En la fase de testing, los datos RSS recibidos desde una posición desconocida son analizados por el clasificador para estimar la posición más cercana. La diferencia entre los algoritmos de reconocimiento de patrones, son en la manera en que utilizan técnicas para emparejar los datos. La salida de estos algoritmos es generalmente un conjunto de probabilidades o verosimilitud de muchas posiciones, con lo cual es posible estimar un centroide de todas las posiciones como solución, o simplemente elegir el que posea la mayor probabilidad.

3.2.3.3. Avances recientes en Fingerprint

La presente sección pretende mostrar los últimos avances en las técnicas de Fingerprint con el objetivo de esclarecer las ventajas y desventajas de estos. Antes, la Figura 3.11 presenta un diagrama de cómo funciona Fingerprint en su forma general:

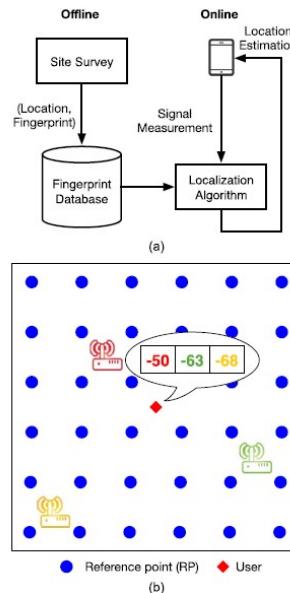


Figura 3.11: (a) Flujo básico de fingerprint. (b) radio-map de un sistema de localización indoor

(Fuente: ([He y Chan, 2016](#)))

1. **Explotando patrones temporales y espaciales:** El Fingerprint tradicional se basa en la medición de vectores RSS. Debido al ruido en las mediciones, el objetivo puede ser *mapeado* a una posición distinta de vectores de señales similares (Sun et al., 2013). La precisión puede ser aumentada si son considerados patrones espaciales y temporales, característicos del lugar en el cual se está experimentando.

- **Patrones temporales** Son patrones de señales que son encontrados a medida que se camina por el área en donde se desea posicionar. Estos patrones pueden ayudar a mejorar la localización en interiores. Luego, cuando un vector de señales es comparado en una posición fija, los patrones contienen información temporal que pueden ser utilizados para restringir y corregir las señales RSS para la localización base.

- **Patrones espaciales** Están caracterizados por asociar señales RSSI con su correspondiente distribución geográfica. Mientras que los patrones temporales requieren conocer el movimiento del usuario, el cual puede ser no determinado o poco preciso en casos reales. Los patrones geográficos de señales pueden ser utilizados para restringir la posición del usuario. Estos patrones incluyen orden de los vectores RSSI, puntos de control en las señales, y la cobertura de las señales dadas por los APs.

2. **Localización colaborativa entre móviles** Muchos de los avances se enfocan principalmente en determinar la posición de un usuario independientemente de los demás dispositivos, sin considerar sus posiciones relativas. Debido a los errores de cada móvil independiente, dos objetivos físicos cercanos pueden ser marcados en diferentes posiciones estimadas. Entonces, si la información de la posición relativa puede ser accedida y utilizada, los resultados en la estimación pueden ser mejorados para la localización individual de cada móvil. Trabajos recientes han desarrollado la localización colaborativa. Estas aristas nacen básicamente de las siguientes tendencias en computación móvil:

- **Contexto de localización de interacción social:** En la localización indoor, la gente puede estar cercana en escenarios sociales típicos (Chan et al., 2006). En museos, por ejemplo, la gente busca dentro junto a su familia o amigos. La interacción entre este conjunto determina un patrón de localización.

- **Dispositivos móviles penetrantes y sensores avanzados:** En estos días, los *smartphones* han desarrollado muchos sensores, los cuales pueden detectar otros móviles en su vecindario cercano, basado en uno o varios protocolos, como puede ser Bluetooth, WiFi direct, NFC y ondas de sonido. El aumento de los dispositivos móviles con estos sensores provee la posibilidad de mejorar significativamente esta técnica en posteriores investigaciones.

3. **Localización asistida por movimiento:** La localización asistida por movimiento es una de las técnicas híbridas más utilizadas en estos días, debido a la gran penetración en el mercado de dispositivos que presentan sensores en el ámbito móvil. Los más grandes avances consisten en las siguientes dos técnicas:

- **Avances en la medición de movimiento:** El monitoreo del comportamiento de una caminata es importante para la localización asistida en movimiento precisa. El mayor desafío en obtener información del movimiento es que los sensores iniciales de los dispositivos móviles sufren de calibración imperfecta y mediciones con ruido. *Step counting* es el mayor acercamiento para capturar el movimiento y dirección de una caminata (Harle, 2013). Trabajos recientes apuntan a mejorar la dirección de la caminata, conteo de pasos y longitud de pasos. Sin embargo, como adaptar estos parámetros a cada usuario, es decir, sus propios perfiles de movimiento es un área desafiante que continua en investigación.

- **Modelos de fusión avanzados y eficientes:** Como fusionar el movimiento con y la detección de señales RSS es esencial para mejorar la precisión en la localización en interiores. El modelo usado en fusión necesita capturar la correlación (ya sea temporal o espacial) entre las señales medidas. Sin embargo, si el modelo es altamente complejo, el alto costo computacional también afecta la calidad de la estimación en la localización. A pesar de esto, buscar un algoritmo de fusión eficiente y preciso se ha convertido recientemente en una importante tendencia para la localización asistida por movimiento (Xiao et al., 2014).

3.3. Motivación de la selección de Fingerprint

Ya que se muestran la mayor parte de las técnicas y tecnologías desarrolladas para la resolución del problema de posicionamiento indoor, a continuación, se describe el motivo de elección de Fingerprint. En primer lugar, es claro que el estado del arte refleja que los modelos más avanzados y de mejores resultados apuntan hacia las tecnologías *Wireless*, vale decir, aquellas que utilizan ondas de radiofrecuencia, por lo cual este tipo de técnicas son las apropiadas para el problema contemporáneo de posicionamiento en interiores, esto se debe a que este tipo de técnicas son muy escalables, requieren poca configuración a priori y además sus costos son relativamente bajos, comparados a otras técnicas por ejemplo basado en visión o en sonido.

En general, como la mayor parte de las tecnologías de radiofrecuencia han sido ampliamente exploradas, como WiFi y UWB por ejemplo, en este trabajo se utiliza Bluetooth, ya que está en constante desarrollo, a tal punto que al momento de escribir esta memoria, se desarrolla la versión 5 de su protocolo, con menos consumo energético y mayor alcance y estabilidad. Además, sus costos son muy bajos en comparación a otras tecnologías y su implementación igualmente, mediante *Beacons* de pequeño tamaño.

La técnica a utilizar es Fingerprint, y esto se debe a que es la que presenta mejores resultados por ejemplo al ser utilizada con modelos estadísticos, o filtros apropiados de ruido. Aunque Fingerprint si ha sido explorado para ser utilizado con algoritmos de machine learning, en su mayoría los estudios son utilizando WiFi, y no se abarca una comparativa de todos los algoritmos recientes de machine learning. Es por ello que, ante la falta de un estudio completo en este tipo de problemas, esta memoria estudia los resultados obtenidos por Fingerprint, mediante la utilización de Bluetooth Low Energy, junto a algoritmos contemporáneos de machine learning, como por ejemplo deep learning. Se deben comparar, evaluarlos y seleccionar los mejores, además de utilizar modelos para reducir el ruido o reducir dimensionalidad.

4 | Propuesta de solución

La propuesta detallada a continuación tiene como consideración los siguientes objetivos referentes a este trabajo:

- Establecer un marco de trabajo para la recolección, entrenamiento y clasificación de algoritmos de machine learning utilizando Bluetooth Low Energy
- Comparación de diferentes clasificadores, para determinar sus ventajas y cuales se comportan de mejor forma en la fase de entrenamiento y test, para así seleccionar los mejores.
- Utilizar técnicas de reducción de dimensionalidad para reducir la complejidad del problema, y optimizar los recursos para así disminuir el procesamiento de los dispositivos móviles.
- Determinar los mejores valores para el funcionamiento correcto de los dispositivos Bluetooth en el contexto del posicionamiento en interiores.
- Utilizar modelos sin necesidad de conexión a internet, es decir, solo utilizar redes Bluetooth sin siquiera tener habilitado el WI-Fi en dispositivos móviles o las redes LTE. Todo el procesamiento se realiza en el cliente, por lo que es importante utilizar algoritmos de gran desempeño y bajo procesamiento.

4.1. Consideraciones Previas

Para resolver el problema de la localización abarcado en esta memoria, hay muchas posibilidades las cuales presentan sus ventajas e inconvenientes con respecto a consumo de recursos, precisión, exactitud, tiempo de procesamiento, entre otros. Por lo mismo, para generar el *framework* de localización, se requiere determinar las opciones factibles que presenten un equilibrio entre todos los parámetros anteriormente nombrados.

Como el objetivo principal de esta memoria es determinar que tan bien se comporta la tecnología *Bluetooth Low Energy* desempeñando la labor de localización, en primer lugar es necesario comparar las diferentes alternativas existentes en el mercado relativas a esta tecnología, que presenten facilidad de uso, un precio relativamente módico y acceso a plataformas de desarrollo o *software development kit* (SDK). A continuación, se detallan las principales características de los Beacons Bluetooth y posteriormente se realiza una comparativa entre los dos proveedores más importantes del mercado.

4.1.1. Definición de Beacons Bluetooth y sus protocolos

La definición más básica de Beacon Bluetooth hace referencia a un dispositivo transmisor de ondas de radio Bluetooth. Estos equipos continuamente emiten señales que otros dispositivos pueden recibir mediante sensores adecuados. Lo que envían corresponde a señales de letras y números en forma de paquetes que se transmiten en intervalos regulares de aproximadamente 100 milisegundos. Dentro de los Beacons existen placas madres que contienen una CPU, transmisor de radio, y baterías. Además, pueden presentar acelerómetro, sensores de temperatura, entre otros.

La transmisión corresponde a un ID único que está presente en cada Beacon y que no se repite, como una dirección MAC o un UUID pertinente, dependiendo de cada fabricante. Este ID por si solo carece de sentido, ya que solo es una dirección, la manera más práctica en que son utilizados estos dispositivos es generar una infraestructura completa y asociar estos ID a posiciones o maquinarias para ser localizadas. Todo esto depende del uso y contexto en que se despliega la aplicación que utiliza los Beacons. Por lo mismo, todo lo que el Beacon transmite es utilizado por los programadores para sus propias necesidades, como enviar un mensaje o anuncio a un smartphone específico, o generar localización en tiempo real.

A pesar de que esta tecnología existe hace muchos años, últimamente se ha vuelto sumamente relevante por el auge del *internet of things*(IOT). Con receptores Bluetooth en 90 % de los equipos móviles, esta tecnología es perfecta para emitir mensajes en un rango corto-medio y generar una completa infraestructura o ecosistema de redes conectadas en cualquier lugar del planeta. Además, desde el 2010, el estándar corresponde a BLE o Bluetooth Low Energy, el cual es una versión mucho más eficiente energéticamente y que ha hecho posible el auge de los Beacons, ya que gracias a este estándar, solo se requiere pequeñas baterías para que estos dispositivos puedan emitir señales por meses e incluso años, lo que convierte a BLE en el mayor avance en IOT por su simplicidad y eficiencia.

4.1.1.1. ¿Como se comunican los Beacons?

Los Beacons envían sus ID únicos aproximadamente 10 veces por segundo dependiendo de la configuración, y cualquier dispositivo que se encuentre en su rango de alcance es capaz de reconocer este ID, como por ejemplo un teléfono celular o un computador portátil. Cuando una aplicación dedicada reconoce este ID, puede emitir un evento en el sistema operativo, como por ejemplo mostrar un mensaje, descargar un archivo de la web, o realizar un algoritmo de localización.

4.1.1.2. ¿Dónde comenzó la tecnología Beacon?

Los Beacons como se conocen hoy en día comenzaron con la creación de los reconocidos iBeacons, implementados por la compañía Apple. iBeacon es un protocolo simple que permite transmitir muy pequeñas porciones de datos. Posteriormente Google lanza su propio protocolo en 2015, denominado EddyStone, como una alternativa a iBeacon. A raíz de esto se ha generado una competencia que ha ayudado a mejorar el hardware y software de los dispositivos Beacons. A continuación, la Tabla 4.1 muestra los parámetros básicos de cualquier dispositivo Beacon, que ayudan a mejorar su eficiencia sin pérdida de precisión.

Tabla 4.1: Parámetros por defecto y métricas esperadas en los dispositivos Beacons Bluetooth

Intervalo	Tx Power	Rango esperado	Batería esperada
100ms	3(-12 dBm)	35m	Hasta 7 meses
300ms	3(-12 dBm)	35m	Hasta 2 años
1000ms	3(-12 dBm)	35m	Hasta 4 años

Los parámetros de la tabla anterior se definen a continuación:

- **Batería esperada:** La mayoría de los Beacons tienen una duración esperada en su batería de 18 a 24 meses, sin embargo dependiendo de las configuraciones y usos su batería puede caer de 6 a 8 meses. Los Beacons con ahorro de energía pueden incluso llegar a 5 años de duración. La explicación en cómo pueden durar tanto, se basa en el protocolo Bluetooth Low Energy el cual es realmente eficiente.
- **Formato soportado:** Habitualmente los Beacons soportan ambos protocolos, es decir iBeacon y Eddystone.
- **Intervalo:** Representa que tan seguido los Beacons transmiten el mensaje. A pesar de que este número puede ser sumamente bajo, la mayor parte de los sistemas operativos no permiten leer tan rápidamente con lo cual un valor menor a 100ms es innecesario en la mayoría de los casos.

- **TX Power:** Este valor describe la potencia de salida, es decir, que tan lejos puede llegar la señal emitida por la radio del Beacon. Puede ser tan pequeña como 4 metros, pero habitualmente pueden llegar a alcanzar valores entre los 50 a 90 metros de distancia con una línea de visión directa.
- **Paquetes transmitidos:** Un paquete Beacon es literalmente los datos que son transmitidos, por ejemplo iBeacon transmite un paquete según su protocolo, mientras Eddystone puede llegar a transmitir 3 de estos simultáneamente.

4.1.1.3. ¿En donde pueden ser utilizados los Beacons?

Los casos de uso más relevantes se listan a continuación:

- **Seguimiento:** En manufactura y transporte es muy relevante llevar el seguimiento de las mercancías o maquinarias en todo momento, para facilitar inventario o saber tiempos de entrega. Adjuntando un Beacon a cada equipo esta tarea es posible y también se puede generar un historial de las locaciones en donde estuvo el equipo.
- **Navegación:** Crear sistemas de posicionamiento indoor es un tema sumamente relevante estos días, ya que GPS no funciona en interiores, mediante infraestructuras basadas en Beacons se puede ayudar a la localización de los usuarios en recintos cerrados.
- **Interacción:** Los Beacons pueden disparar eventos en los dispositivos móviles de los usuarios u otros equipos, como generar notificaciones, activar equipos como luces o televisores, etc.
- **Análisis de datos:** Mediante los dispositivos Beacon se puede establecer una completa base de datos por ejemplo donde los usuarios compran más, o que lugares son mayormente transitados en un recinto comercial, también puede ayudar a detectar fallos en líneas de producción antes que ocurran y guardar todos estos datos en la nube para posteriormente ser analizados.

4.1.1.4. Perfiles Beacon

- **iBeacon:** Este protocolo es el primero y más ampliamente difundido. Desarrollado por Apple es nativamente soportado por IOS. Además, también es soportado en otros sistemas operativos móviles, pero funciona de mejor forma en iPhone y iPad. La manera en que funciona este protocolo corresponde a una combinación de letras y números separados en grupos específicos. Un paquete iBeacon se compone de los siguientes elementos:

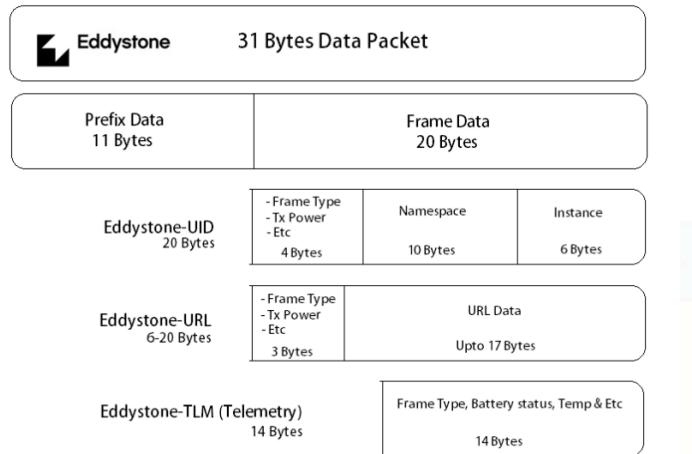
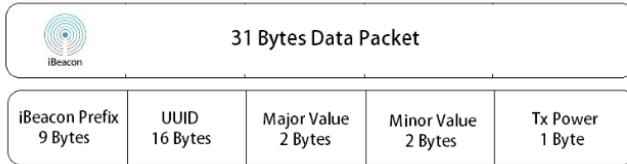
Unique Universal Identifier(UUID): La información más general del Beacon, por ejemplo el edificio al cual corresponde el Beacon.

Major: La información espacial más general del Beacon, por ejemplo un Beacon que pertenece a una determinada tienda dentro del edificio.

Minor: Una pieza de datos más pequeña, representa por ejemplo una estantería o mesa específica.

- **Eddystone:** Este protocolo es inicialmente desarrollado por Google específicamente para dispositivos Android, permitiendo así ser más interoperables y de código abierto. La manera en que funciona el protocolo Eddystone es similar a iBeacon, pero extienden su funcionalidad de la siguiente forma, enviando 4 paquetes de diferentes tipos: Eddystone-UID, Eddystone-URL, Eddystone-TLM y Eddystone-EID. Eddystone-UID funciona prácticamente como lo hace iBeacon. Eddystone-URL envía una URL a los dispositivos móviles permitiendo abrirla instantáneamente en el navegador sin necesidad de utilizar una aplicación. Eddystone-TLM envía datos telemétricos y de sensores asociados al mismo Beacon. Para Eddystone-UID los Beacons envían una parte denominada NameSpace y otra llamada Instance, las cuales son similares a Major y Minor de iBeacon. Eddystone puede abarcar cualquier paquete mencionado anteriormente, pero añade una capa de seguridad, cambiando su ID constantemente.

La Figura 4.1 y 4.2 muestran como son construidos los paquetes de cada protocolo y su respectivo tamaño en bytes.

**Figura 4.1:** Tipos de paquetes Eddystone con su respectivo tamaño.(Fuente: ([Ramananda Shetty, 2015](#)))**Figura 4.2:** Paquete iBeacon realizado por Apple.(Fuente: ([Ramananda Shetty, 2015](#)))

4.1.2. Estabilidad de la señal Bluetooth

Las señales u ondas de radio son altamente afectadas por ruido o interferencia presente en el entorno, perdiendo así intensidad en la señal o resultando en valores fluctuantes que causan estimaciones o eventos no esperados. Esto ocurre por ejemplo en las interferencias de señales Wi-Fi al descargar un archivo, o también en equipos Bluetooth al conectarse a un dispositivo de música. La localización en interiores al depender igualmente del valor de *Received Signal Strength Indicator*(RSSI) o intensidad de la señal recibida; se ve ampliamente afectado, sobre todo por el hecho de que los usuarios están caminando y en un entorno cambiante. El RSSI habitualmente se expresa en dBm para medir la potencia obtenida por una señal recibida. Su valor mayor teórico es 0 dBm y la escala corresponde a valores negativos, en donde más negativo implica una mayor pérdida de señal.

El cuerpo humano, así como otros objetos, provoca una pérdida de señal debido a la atenuación de esta, debido a la reflexión o refracción que ocurre según los materiales en donde la señal incide. El cuerpo humano provoca absorción de las señales y disminución de la calidad de estas. Además, la propagación multicaminos es un efecto latente en todas las ondas de radio, en donde las ondas llegan al receptor por diferentes caminos y diferentes tiempos, provocando interferencia constructiva o destructiva. Este fenómeno habitualmente ocurre por los medios en donde se transportan las ondas y el entorno como tal, provocando que la señal obtenida difiera de la original manifestando ruido e interferencia, disminuyendo así la calidad e intensidad de esta.

Para corroborar como afectan estos fenómenos a la intensidad de la señal recibida por Beacons Bluetooth, se realiza una prueba en donde se pueden visualizar los efectos de obstrucción en la señal. Para realizar esta prueba se utiliza un teléfono celular con una pequeña aplicación la cual es capaz de percibir las señales Bluetooth emitidas por un dispositivo Beacon en sus cercanías. Cada cierto tiempo se registran los valores RSSI percibidos y se escriben en un archivo de texto para posteriormente ser procesados. Se mide

la señal durante 3 minutos con una línea de visión limpia, es decir, sin obstrucciones o interferencias entre ambos dispositivos. Luego, una persona camina entre la línea de visión y también permanece quieto en ella, obstruyéndola completamente. La Figura 4.3 muestran los resultados de estas pruebas.

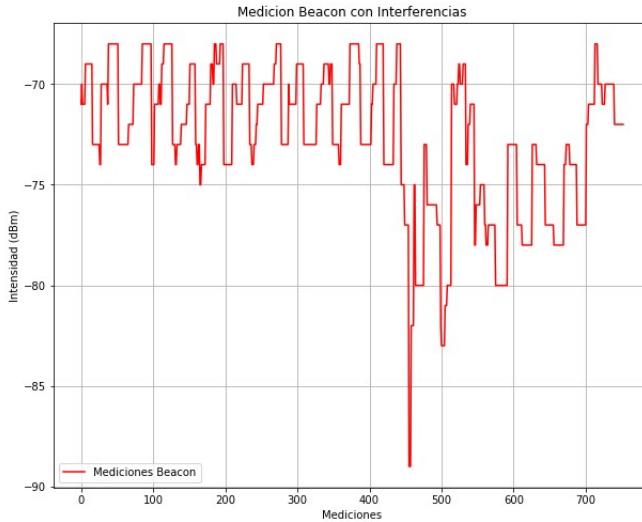


Figura 4.3: Resultados del cambio en la señal producto de una persona caminando delante de la línea entre un Beacon y un teléfono celular receptor
(Fuente: Elaboración propia)

Como se observa en la imagen, en un principio, la señal oscila en valores inferiores a -75 dBm, lo cual es aceptable y medianamente bueno según la escala RSSI. Posteriormente a partir de la medición 400, se introduce la aparición de la persona en la línea de propagación de la señal, con lo que inmediatamente la intensidad de la señal se reduce significativamente, incluso llegando a los límites de -90 dBm que es básicamente inestable, ya que para una conexión segura al menos se recomienda -80 dBm. Posteriormente la persona se mueve a través de la línea de visión, provocando los efectos de la gráfica, es decir, señales intermitentes y poco estables.

Todo esto demuestra principalmente que no se puede confiar completamente en el valor de la intensidad de la señal recibida, ya que es muy probable que arribe con efectos de multicamino u obstrucciones que modifican su valor durante el viaje. Este efecto es significativamente mayor en grandes distancias de viaje en las señales, lo cual sugiere que los Beacons instalados no tengan una disposición tan distante, es decir, para obtener mejores resultados es mucho mejor una gran densidad de Beacons por superficie, lo que aumenta el costo de despliegue. Por otra parte, los algoritmos de posicionamiento deben aprender a leer e interpretar este ruido, basado en patrones según la posición, con lo cual los algoritmos supervisados de máquinas de aprendizaje son sumamente efectivos en este ámbito.

4.1.3. Evaluación y selección de Beacons Bluetooth

A pesar de que en el mercado existe una gran variedad de empresas proveedoras de Beacons Bluetooth, hay dos que se destacan por sobre los demás y que son pioneras dentro de esta industria, estas son Estimote ([Estimote, 2017](#)) y Kontakt ([Kontakt, 2017](#)). Ambas empresas proveen de hardware y soluciones de software ya implementadas y listas para el uso empresarial o pequeños desarrollos. Para compararlos solo se consideran los Beacons estándar dentro de cada proveedor, ya que existen múltiples variedades en cada segmento, sin embargo, estos poseen otras cualidades como sensores de luz, temperatura, lectores NFC, UWB, o son resistentes al agua, polvo, entre otros. Para hacer un análisis justo y conciso, en este caso se obviarán estas características, aunque si pueden ser utilizadas en diversos contextos.

Para la comparativa se utilizan las características indicadas por cada fabricante y estas se resumen en la Tabla 4.2:

Tabla 4.2: Tabla comparativa Estimote y Kontakt.IO

Parámetro	Kontakt.io	Estimote
Duración de la batería	Hasta 4 años	Hasta 2 años
Rango	70m	70m
Procesador	32-bit ARM® Cortex™ M0 CPU core	ARM® Cortex®-M4 32-bit processor FPU
Sensibilidad	-93dBm	-96 dBm
Velocidades	250kBps, 1Mbps, y 2Mbps	1 Mbps (2 Mbps soportado)
Memoria	256KB flash 16KB RAM	512 kB Flash memory 64 kB RAM memory
Transmission power	-30dBm to 4dBm	-20 to +4 dBm
Batería	2 x 1.000mAh CR2477	1 x CR2477 – 3.0V
Bluetooth	Bluetooth® 4.2 LE standard	Bluetooth® 4.2 LE standard
Espesor	15mm	17mm
Peso	35 gr	30 gr
Paquete iBeacon y Eddystone	1 a la vez	1 a la vez
Paquetes adicionales	telemetría	telemetría
Sensores adicionales	Temperatura	movimiento, temperatura
Batería reemplazable	Si	Si
Número de Beacons	3	3
Precio	60 USD	59 USD

La Tabla 4.2 muestra que ambos equipos son muy similares en teoría, además ambos presentan kits de desarrollo de software para las principales plataformas, ya sea web, aplicaciones móviles como Android e IOS como también permiten la integración de la infraestructura en la nube. Para determinar entonces la elección, es necesario realizar pruebas en ambos hardware, sin embargo, estas pruebas ya han sido realizadas en múltiples ocasiones por investigadores como se aprecia en ([Florian Trautmann, 2015](#)) en donde se realizaron pruebas de fluctuación en las señales para ambos equipos. La prueba consiste en poner Beacons Kontakt y Estimote a una distancia de 2 metros de un dispositivo móvil que registra la señal. Ambas configuraciones son idénticas en los equipos y las mediciones fueron realizadas cada 500ms. Los resultados de las pruebas indican que los Beacons de Kontakt tienen muchas menos fluctuaciones y la señal permanece estable por mucho más tiempo, lo que es un indicio que pueden ser mucho más factibles en la localización en tiempo real.

4.1.4. Algoritmos de *Machine Learning*

Para determinar la posición del dispositivo móvil, es necesario contar con un algoritmo que pueda inferir la posición del usuario basado en las señales percibidas por los distintos Beacons. Para ello en este trabajo se utilizarán los denominados algoritmos de aprendizaje automático o *machine learning*. En términos generales los algoritmos de aprendizaje automático corresponden a una rama de la inteligencia artificial en donde su objetivo principal es lograr que los computadores puedan aprender por sí mismos, sin necesidad de programar su comportamiento explícitamente. Para lograr esto, se entrena los diferentes algoritmos mediante información suministrada en forma de ejemplos según el tipo de resultado que se espera. El auge de este tipo de técnicas ocurre con la llegada del internet y los grandes volúmenes de datos, ya que para resolver ciertos problemas como clasificación de imágenes, los algoritmos convencionales son NP-Hard, es decir, su complejidad computacional es muy alta o desconocida, por lo tanto no es computacionalmente computable en un tiempo razonable, por lo que las técnicas de aprendizaje automático o heurísticas pueden entregar resultados positivos con una alta precisión sin necesidad de programar un algoritmo específico a cada problema.

Para la utilización de estos tipos de técnicas, es necesario una representación adecuada de los datos acorde al problema que se requiere resolver, para ello se deben describir los datos, por ejemplo, los pixeles de una imagen como una matriz de atributos, es decir, cada registro o fila es una imagen y cada atributo, campo

o columna de la matriz es un pixel de la imagen. La representación de los datos es de vital importancia para obtener unos resultados positivos.

Dentro del aprendizaje automático se distinguen dos tipos de técnicas de las más conocidas, las cuales son aprendizaje supervisado y no supervisado. El aprendizaje supervisado corresponde a deducir una función o mapeo según los datos con los cuales es entrenado el modelo, para ello al momento de entrenar se debe suministrar pares de objetos, en donde una componente son los datos de entrada, por ejemplo, los pixeles de una imagen; y en la segunda componente se suministra la clase de esta imagen, que puede corresponder por ejemplo al objeto presente en la imagen, como un tipo de animal o automóvil, dependiendo del problema a tratar. Entonces la función es capaz de determinar un valor numérico (regresión) o una etiqueta (clasificación) para nuevos datos no vistos anteriormente. Por lo anterior, es necesario que el modelo generalice adecuadamente y no se sobre ajuste a los datos de entrenamiento, es decir, solo sea capaz de inferir correctamente sobre el conjunto de entrenamiento y no para nuevos datos. Mientras más datos son suministrados, más fácilmente es para las técnicas de aprendizaje automático predecir correctamente para nuevos datos.

Estos algoritmos son típicamente implementados en dos fases. En la primera fase, denominada fase de entrenamiento o **training phase**, los datos son recopilados y manipulados de forma tal que pueden ser suministrados a los algoritmos de aprendizaje automático, de esta forma, los algoritmos pueden “aprender” patrones de los datos y clasificarlos. En la segunda fase, denominada fase de pruebas o **testing phase**, nuevos datos no vistos previamente son probados en el algoritmo entrenado en la primera fase, para dilucidar la efectividad del modelo construido.

A lo largo de este trabajo se utilizan muchos tipos de algoritmos, sin embargo, a continuación, se procede a describir los más relevantes y que serán mayormente utilizados en la localización en interiores, ya que a lo largo de investigaciones previas han demostrado buenos resultados en problemas similares. La Figura 4.4 muestra el esquema básico que sigue cualquier técnica de machine learning.

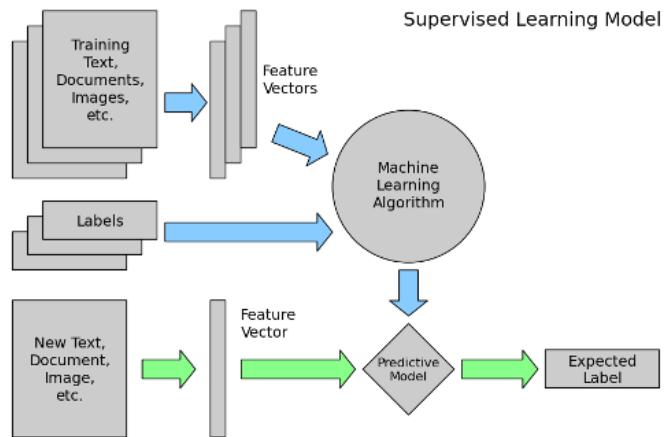


Figura 4.4: Esquema básico del funcionamiento de las técnicas de aprendizaje automático

(Fuente: [Maribel Tirados \(2014\)](#))

4.1.4.1. K-Nearest Neighbor

El método K-Nearest Neighbor o método de los **k** vecinos más cercanos, abreviado también como **k-nn** es un método de clasificación y regresión supervisada, muy simple, pero a la vez poderoso, ya que es no paramétrico, es decir, no es necesario configurar parámetros del modelo, por lo que entradas de datos similares deben tener salidas de datos similares. Este método estima la función de densidad $F(\frac{x}{C_j})$ de las predictoras x (datos de entrada) por cada clase C_j . En el proceso de aprendizaje no se hace ninguna suposición sobre la distribución de las variables predictoras.

Aunque es un algoritmo supervisado, su fase de entrenamiento es muy acotada, enfocándose mayormente en la fase de pruebas, ya que todo el esfuerzo realizado por este algoritmo es hecho sobre la marcha, es decir, mientras se ejecuta estima la mejor predicción en cada instante de la fase de entrenamiento, por lo que también se habla de un algoritmo de aprendizaje basado en instancias.

Los datos son representados por vectores en un espacio multidimensional, en donde cada ejemplo posee p atributos y existen q clases para clasificarlos. Luego el algoritmo esencialmente separa el espacio multidimensional en regiones, en donde un nuevo dato será clasificado como perteneciente a una clase C siempre y cuando esta clase sea la más frecuente entre los k vecinos más cercanos al dato en cuestión. Para medir la distancia entre datos o vectores, se puede utilizar distancia euclídea, manhattan, Chevyshev, distancia del coseno, entre otras. Luego, por ejemplo, para la distancia euclídea se tiene la siguiente fórmula:

$$d(x_i, x_j) = \sqrt{\sum_{r=1}^p (x_{ri} - x_{rj})^2} \quad (4.1)$$

Luego, un dato está descrito como:

$$x_i = (x_{1i}, x_{2i}, \dots, x_{pi}) \in X \quad (4.2)$$

Luego la fase de entrenamiento del algoritmo consiste en almacenar los valores y sus características, así como las etiquetas. En la fase de clasificación, se mide la distancia entre el nuevo vector y los datos de entrenamiento previamente almacenados y se seleccionan los k ejemplos más cercanos, para posteriormente ser clasificado según la clase que más se repite dentro del conjunto de elementos seleccionados.

Este algoritmo tiene un problema al suponer que todos los atributos son igualmente relevantes, ya que, al usar la distancia euclídea, todos valen lo mismo, por lo que un atributo relevante tiene el mismo “peso” que uno irrelevante. Sin embargo, este problema no ocurre en la clasificación utilizando Beacons, ya que cada una de las señales emitidas por distintos Beacons es igual de relevante y no importa su peso.

Un problema importante al tratar con algoritmos basados en distancia es el llamado efecto de la “maldición de la dimensionalidad” o *curse of dimensionality*, el cual hace referencia al problema de encontrar patrones en espacios de altas dimensiones, ya que mientras más alta es la dimensionalidad de los datos, más dispersos están y para seguir obteniendo resultados buenos, es necesario suministrar una cantidad de datos que crece exponencialmente. Además, al ser tan grande el espacio dimensional, es mucho más complejo encontrar patrones o regiones para lograr una correcta clasificación.

Por último, al tener muchos atributos, y muchos datos de entrenamiento, k-nn se torna muy lento, ya que para inferir un nuevo dato de prueba, requiere recorrer todos los datos de entrenamiento, provocando que el tiempo de procesamiento aumente significativamente a medida que aumenta el número de datos en el *dataset*. Esto es un punto relevante en el posicionamiento indoor, ya que en un simple despliegue de una infraestructura de Beacons por ejemplo en un centro comercial, se deben almacenar miles de puntos de referencia, con lo cual k-nn puede reducir su *performance* y demorar demasiado, lo cual es crítico en posicionamiento en tiempo real.

La Figura 4.5 muestra el funcionamiento de k-nn y como el valor seleccionado de k puede afectar en la clasificación.

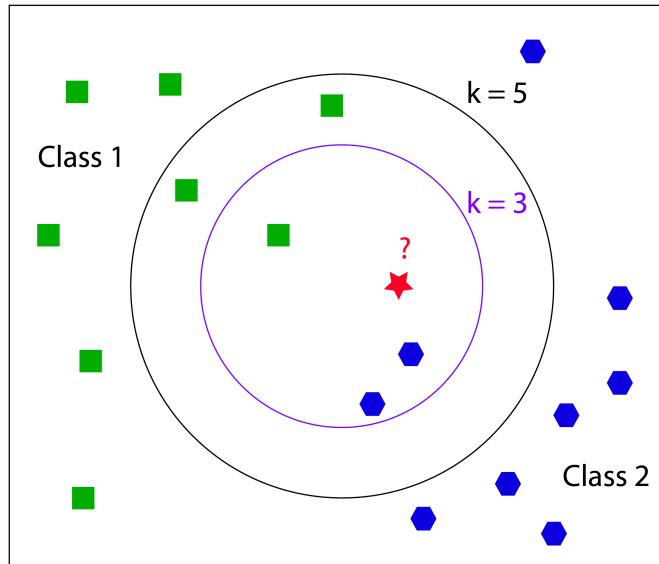


Figura 4.5: Ejemplo de clasificación utilizando k-nn con $k = 5$ y $k = 3$

4.1.4.2. Support Vector Machines

Otro de los algoritmos más utilizados en aprendizaje supervisado corresponde al denominado *support vector machine*(SVM) o máquina de vectores de soporte. Ese algoritmo es sumamente utilizado sobre todo en problemas de clasificación, regresión y *detección de outliers*. Una máquina de vectores de soporte construye un hiperplano o un conjunto de ellos, en un espacio de alta dimensionalidad o dimensionalidad infinita, el cual puede ser utilizado para clasificación. Intuitivamente, un buen hiperplano separador es aquel que logra maximizar la distancia a cada ejemplo de entrenamiento más cercano a él, de cada clase, lo cual se conoce como margen funcional, entonces mientras más grande es este margen, menor es el error de generalización del clasificador, es decir, permite mayor generalización para nuevos datos, sin definir un margen estricto que solo se adapte a los datos de entrenamiento. Posteriormente para clasificar nuevos ejemplos, la SVM ubica este nuevo punto en una región del espacio que corresponde a una clase, según su distancia y posición respecto al hiperplano definido en la fase de entrenamiento.

Como muestra la Figura 4.6, este hiperplano logra una separación en un espacio de dos dimensiones, lo cual es adecuado en clasificación binaria. Aunque la formulación matemática es idéntica, para resolver SVM en problemas de multclases, es necesario escoger un esquema de resolución. Una de las aproximaciones más utilizadas de resolución es reducir el problema de multclasificación en muchos problemas de clasificación binaria. Para ello se utilizan formulaciones conocidas en la inteligencia artificial como *one-vs-all* o esquemas de votos como *one-vs-one*. La formulación matemática se muestra a continuación:

Dado un conjunto de vectores de entrenamiento $x_i \in \mathbb{R}^p$, $i = 1, \dots, n$ en dos clases y un vector $y \in \{1, -1\}^n$, SVM resuelve el siguiente problema de optimización:

$$\min_{w,b,\zeta} \frac{1}{2} w^T w + C \sum_{i=1}^n \zeta_i \quad (4.3)$$

sujeto a:

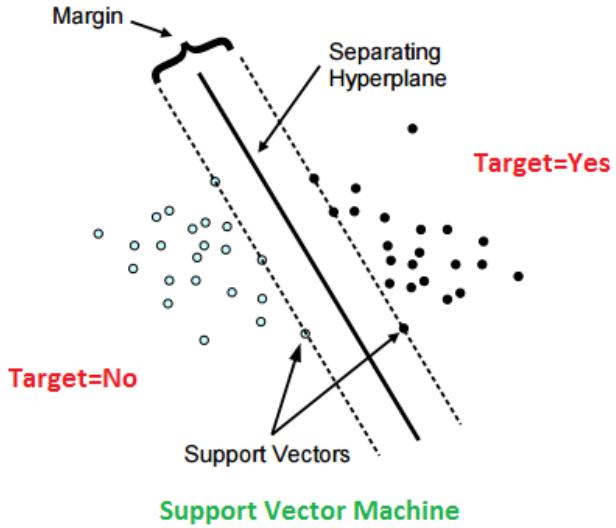


Figura 4.6: Ejemplo SVM en dos dimensiones, en donde se muestran los vectores de soporte y el hiperplano que logra la máxima separación

$$y_i(w^T \phi(x_i) + b) \geq 1 - \zeta_i, \quad (4.4)$$

$$\zeta_i \geq 0, \quad (4.5)$$

$$i = 1, \dots, n \quad (4.6)$$

En su forma dual de problema de programación lineal, este problema es puede ser expresado como:

$$\min_{\alpha} \frac{1}{2} \alpha^T Q \alpha - e^T \alpha \quad (4.7)$$

sujeto a:

$$y^T \alpha = 0, \quad (4.8)$$

$$0 \leq \alpha_i \leq C, i = 1, \dots, n \quad (4.9)$$

En donde e es el vector de unos, $C > 0$ es el límite superior, y Q es una matriz semidefinida positiva de tamaño $n \times n$, $Q_{ij} = y_i y_j K(x_i, x_j)$ en donde $K(x_i, x_j) = \phi(x_i)^T \phi(x_j)$ se denomina el kernel y es sumamente importante en SVM, ya que este permite llevar desde un espacio en donde la separación de las clases no es tan clara, a un espacio de dimensiones muy grandes o infinitas, en donde la separación se vuelve mucho más obvia.

Finalmente, la función de decisión corresponde a:

$$sgn\left(\sum_{i=1}^n y_i \alpha_i K(x_i, x) + \rho\right) \quad (4.10)$$

La relevancia del kernel es lo que torna a SVM muy factible en problemas de clasificación, ya que permite llevar los datos de un espacio finito, en donde no son linealmente separables, a un espacio de

dimensión infinita en donde si pueden ser separados linealmente, lo cual es conocido como *kernel-trick*. Además, como solo se utilizan funciones kernel que utilicen producto punto, la operación de llevar estos datos al nuevo espacio es eficiente y de bajo costo computacional. Para realizar el kernel trick, cada producto punto es reemplazado por una función kernel de carácter no lineal. El kernel más utilizado es RBF o radial basis function el cual es definido como:

$$K(x_i, x_j) = \exp(-\gamma \|x_i - x_j\|^2), \gamma > 0 \quad (4.11)$$

Luego, utilizando este kernel, se debe escoger el parámetro γ y el parámetro C adecuado, los cuales son muy importantes en una correcta generalización del SVM. Habitualmente SVM al ser un hiperplano de máxima distancia a cada clase, es muy estricto en la clasificación, lo cual no siempre es lo ideal, debido a que pueden existir *outliers* debido a ruido, afectando a toda la clasificación. El hiperparámetro C ayuda en este sentido, con lo que se denomina *soft-margin*, es decir, un margen más permisivo, que clasifica de manera errónea algunos ejemplos de entrenamiento, buscando la generalización del SVM.

Por otra parte, el hiperparámetro γ define que tan lejos alcanza la influencia de un ejemplo de entrenamiento por sí solo, es decir, puede ser visto como el inverso del radio de influencia de los ejemplos de entrenamiento escogidos como vectores de soporte. Un valor pequeño de γ producirá un largo alcance, definiendo zonas de clasificación muy amplias, con lo cual no se aprecia completamente la forma o patrón de estas. Por otra parte, un valor grande, provoca demasiada separación, aislando cada ejemplo de entrenamiento.

4.1.4.3. Redes Neuronales y Deep learning

Uno de los campos de investigación con mayor auge en el último periodo, es todo lo relativo a los avances en redes neuronales, particularmente *deep learning*. La base de estos modelos computacionales es, como indica su nombre, las neuronas y su funcionamiento en el cerebro y el sistema nervioso. El cerebro, en particular el sistema visual o corteza visual primaria, es el responsable de reconocer patrones, y lo hace de manera tal en que una imagen es dividida y analizada a través de muchas capas, disminuyendo la complejidad en cada una de ellas para ir de lo macro a lo micro, pudiendo así establecer patrones de menor escala y localidades en la imagen que ayuda a distinguir de que se trata el objeto visualizado.

Cada unidad neuronal está conectada a las neuronas de la capa siguiente y ellas pueden aumentar o inhibir el estado de las neuronas en la capa siguiente. Estos sistemas por lo mismo pueden aprender solos, solo se debe suministrar un número adecuado de ejemplos para el entrenamiento.

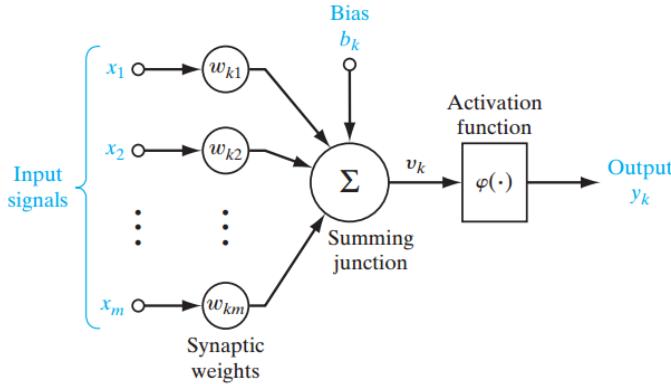
Cada activación de la neurona en la capa siguiente depende de los pesos w y el sesgo o *bias* b . Los pesos multiplican a las entradas en cada conexión, y el bias es el límite, de donde se establece si la suma ponderada de las entradas con los pesos es mayor o menor que el bias, entonces la neurona se activa.

La Figura 4.7 muestra como cada entrada es multiplicada por su respectivo peso, y la suma ponderada se suma con el bias, para luego utilizar la función de activación y obtener la salida, la cual describe que tanto aporta esta neurona, la cual será el nuevo input para la siguiente capa. La neurona perceptron queda descrita como:

$$\text{output} = \begin{cases} 0 & \text{si } w \cdot x + b \leq 0 \\ 1 & \text{si } w \cdot x + b > 0 \end{cases} \quad (4.12)$$

En donde $w \cdot x = \sum_j w_j x_j$ con w y x son vectores en donde sus componentes son los pesos y las entradas respectivamente, y el número de neuronas en la capa corresponde a j . A pesar de que este modelo sirve para problemas simples, no funciona de buena manera en problemas altamente no lineales.

La neurona más utilizada es la denominada neurona sigmoid. Estas neuronas son similares a las neuronas perceptron(primeras neuronas inventadas), pero pequeños cambios en sus pesos y sesgos, provocan

**Figura 4.7:** Esquema básico de single layer perceptron(Fuente: [Syed Danish Ali \(2016\)](#))

pequeños cambios en las salidas, lo cual las hace mucho más útiles y diversifican muy bien la información a través de la red. La función de activación de este tipo de neuronas corresponde a la función sigmoid, y ayuda a resolver problemas estrictamente no lineales. Luego el output y_k viene dado por:

$$\sigma(w \cdot x + b) = \frac{1}{1 + \exp(-\sum_j w_j x_j - b)} \quad (4.13)$$

Luego el procedimiento por el cual aprende la red corresponde a un algoritmo denominado *backpropagation* el cual puede computar los pesos y bias adecuados para la red, con el objetivo de entrenarla y definir los mejores hiperparámetros para posteriormente clasificar de manera correcta nuevos ejemplos. El algoritmo de backpropagation consiste en los siguientes pasos:

1. **Input:** Iniciar la correspondiente activación de la capa de entrada
2. **Feedforward:** Computar las salidas de cada neurona de las capas ocultas. Finalmente computar la salida de la capa de salida.
3. **Output error:** Computar los errores de la capa final o capa de salida
4. **Backpropagation:** Propagar el error de la capa de salida a la capa anterior oculta. Repetir para todas las capas hasta llegar a la capa de entrada.
5. **Output:** Calcular el gradiente de la función de costo, para luego actualizar los pesos y bias.

Para calcular el gradiente se utiliza gradiente descendente, y entonces, cada iteración consiste en atravesar la red hacia adelante, computar el error y volver para propagar el error y computar los ajustes y actualizaciones a los hiperparámetros. Este algoritmo ha logrado un gran avance en las redes neuronales artificiales permitiendo entrenarlas de manera mucho más rápida, logrando grandes resultados.

Las redes neuronales obtienen buenos resultados en gran parte de los problemas, pero para problemas demasiado complejos el concepto de multilayer perceptron, es decir, el modelo estándar de redes neuronales no es suficiente. Para ello, ha surgido el concepto de *deep learning* o aprendizaje profundo. Estas redes son similares a las redes normales, pero poseen más de una capa escondida y además se han desarrollado otros algoritmos para resolverlas eficientemente. La Figura 4.8 muestra una red profunda.

Para resolver este tipo de redes profundas existen muchos acercamientos, y cada día nacen nuevas formas de resolución. Las formulaciones matemáticas son muy extensas, pero en términos generales los algoritmos de backpropagation y gradiente descendente son las bases fundamentales de resolución, a pesar de que necesitan modificaciones para que la red no se quede atascada en ciertas capas, cosa que habitualmente ocurre en redes profundas.

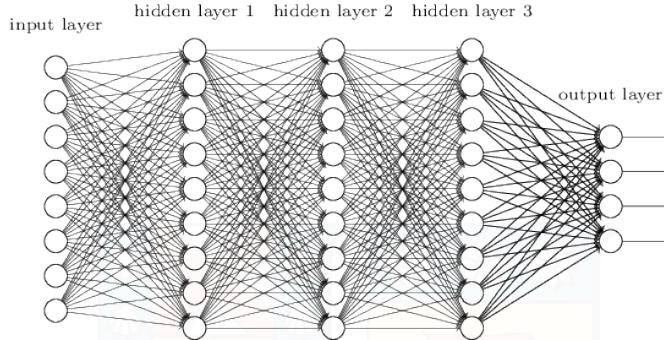


Figura 4.8: Red neuronal profunda con múltiples capas escondidas

(Fuente: [Syed Danish Ali \(2016\)](#))

4.2. Descripción del *framework* de posicionamiento

Para la elaboración del sistema de posicionamiento, se utiliza la técnica de Fingerprint discutida en el estado del arte, la cual, mediante la utilización de un mapa de señales, también denominado *radiomap*, puede inferir la posición de un usuario utilizando algún algoritmo de clasificación. Como el objetivo de este trabajo es determinar los mejores algoritmos de máquinas de aprendizaje para posicionamiento indoor utilizando Bluetooth Low Energy, es necesario establecer un marco de trabajo mediante el cual se pueda llevar a cabo esta tarea.

Lo primero a tener en consideración, es que se deben utilizar dispositivos Bluetooth Low Energy, los cuales realizan la función de access point(AP) y que serán los responsables de emitir la señal RSSI. Luego, el procedimiento se divide en las dos clásicas etapas de Fingerprint, es decir, fase *offline* y fase *online*.

4.2.1. Fase Offline

Para la generación del radiomap, se debe crear un tipo de aplicación que sea capaz de recolectar los vectores RSSI en diversos puntos dentro del lugar de experimentación. Esta aplicación debe ser simple y permitir la adición de nuevos puntos Fingerprint. Por lo tanto, esta es una tarea muy importante, ya que a partir de estos datos se implementarán los algoritmos de máquinas de aprendizaje. El periodo y frecuencia de los datos se debe determinar experimentalmente. Para ello, cada medición a colectar representa un punto en el espacio 2 – dimensional, es decir, un punto dentro del plano. Para generar la grilla, es necesario tener la posición exacta, que corresponde a la etiqueta de cada punto mapeado. Entonces, se formará una grilla de múltiples puntos con sus respectivos fingerprints leídos. Cabe destacar que los puntos de referencia en donde se toman los datos no necesariamente deben estar equiespaciados, pero de esta manera es mucho más simple formar la grilla, ya que cada punto de medición puede corresponder al centro de un cuadrado de determinadas dimensiones. La Figura 4.9 muestra la grilla a desarrollar, con un punto de referencia y su respectivo vector RSSI expresado en dBm.

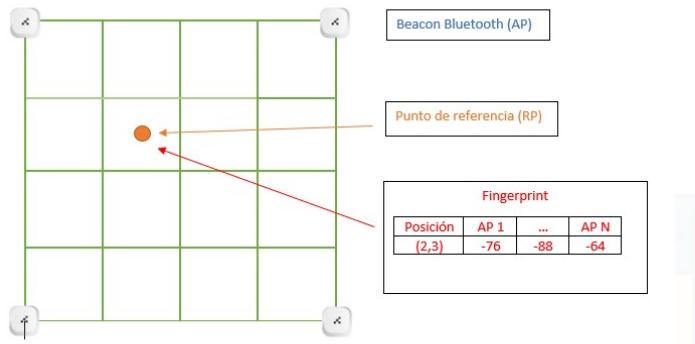


Figura 4.9: Ejemplo de grilla utilizada, con sus puntos de referencia y los Fingerprints leídos, que corresponden a vectores de señales RSSI

(Fuente: Elaboración propia)

Con los datos registrados, se debe crear la base de datos que almacenara estos Fingerprints, ya que desde ahí es posible analizar los datos y mantener su persistencia. Posteriormente, con estos datos se crea el radiomap respectivo, es decir, asociar un vector de señales RSSI, en donde cada componente representa la intensidad recibida proveniente de un access point, a cada posición en donde se colectaron los datos. El número de fingerprints depende del número de posiciones en donde se colectan los datos, y también de cuantos ejemplos se obtienen en cada una de estas posiciones. La Tabla 4.3 ejemplifica la estructura del radiomap a construir:

Tabla 4.3: Estructura general del radiomap a construir

Posicion	AP 1	AP 2	...	AP N
(3,3)	-65	-70	...	-95
(6,6)	-80	-50	...	-76

Cabe destacar que los algoritmos de máquinas de aprendizaje a pesar que pueden resolver para problemas multivariados, es decir, predecir para dos variables, en este caso (x, y) como un punto; es mucho más sencillo separar esto en dos problemas, vale decir, un algoritmo de máquina de aprendizaje para x y otro para y . Otra alternativa sería asociar una clase a cada punto, es decir, numerar estos puntos de la grilla, pero el problema con esto es que se torna mucho más complejo de aprender, debido a que se necesitarían demasiados datos para verdaderamente lograr un aprendizaje de cada una de estas clases, provocando un aumento significativo en la complejidad del problema.

Teniendo en consideración los puntos anteriormente mencionados, el siguiente paso es entrenar los algoritmos de aprendizaje a ser comparados y utilizados en el posicionamiento. Antes del entrenamiento, es necesario de destacar que se utilizará un algoritmo de reducción de dimensionalidad. Esto no ha sido mayormente explorado en la literatura, sin embargo, puede ser de mucha utilidad a la hora del entrenamiento y la fase *online*. La principal razón de utilizar estas técnicas es que los puntos de referencia poseen una alta correlación, debido a que la onda decae según el cuadrado de la distancia, por lo que se presenta una correlación entre dos puntos adyacentes. Por otra parte, los métodos de extracción de características pueden ayudar a agilizar la fase de entrenamiento, ya que este proceso es lento. Además, al ser menos componentes, en la fase online, las técnicas tardarán mucho menos tiempo en determinar la posición en tiempo real, lo cual es sumamente efectivo en técnicas como KNN.

Existen dos métodos muy conocidos mediante los cuales se puede reducir la dimensionalidad, estos son *Linear discriminant analysis (LDA)* y *principal component analysis (PCA)*. Por un lado, LDA es una técnica habitualmente utilizado en los campos de estadísticas, reconocimiento de patrones y máquinas de aprendizaje. El objetivo de LDA es encontrar una combinación lineal de los *features* o características del

dataset, las cuales separan dos o más clases. La combinación lineal resultante es habitualmente utilizada como clasificador o más popularmente para reducir la dimensionalidad del dataset. LDA es similar a PCA, ya que ambos buscan combinaciones lineales que explican de mejor manera los datos, sin embargo, la manera en que los realizan es diferente, ya que LDA explícitamente intenta modelar la diferencia entre las clases de datos, por lo que necesita de las etiquetas de cada clase, lo cual lo torna un método supervisado. Por otro lado, PCA no toma en cuenta la diferencia entre las clases, por lo que es un método no supervisado. Además, LDA funciona bien en el caso de que las variables o atributos del dataset son independientes, y predicen la variable categórica o etiqueta de cada clase.

Como se mencionó anteriormente, LDA maximiza la separación entre clases, es decir, la varianza entre clases o interclases. PCA por su parte, busca las componentes de la combinación lineal, las cuales maximizan la varianza en los datos, esto quiere decir, la varianza de las variables o varianza intraclasses. LDA hace presunciones sobre los datos que lo vuelven más complejo de implementar, por ejemplo, las clases deben estar normalmente distribuidas, y la covarianza de las clases debe ser igual. Por todo lo anterior, y como es necesario reducir la correlación entre los puntos adyacentes, se decide utilizar PCA como técnica para reducir el espacio de dimensionalidad y para encontrar un espacio no correlacionado de variables. Esto es necesario, debido a que al estar tan correlacionado los puntos de referencia en la grilla, puede existir información o *features* escondidos, lo cual no aumentara la *accuracy* de las técnicas de machine learning, pero si mejora los tiempos de entrenamiento y en la fase online, lo cual es muy relevante en un sistema de posicionamiento en tiempo real. La correlación se debe a que como es sabido, las ondas se propagan de la siguiente forma:

$$I \propto \frac{1}{r^2} \quad (4.14)$$

En donde I representa la intensidad y r es la distancia de propagación desde el punto de emisión. Esta fórmula define como decrece la intensidad de la señal a medida que esta se transmite en el medio según el inverso del cuadrado de la distancia. Con lo anterior, y considerando que los Beacons actúan como antenas omnidireccionales, si el RSSI de dos Beacons se mide en un punto, luego se traslada a otro punto adyacente, y se mide nuevamente, al ser la intensidad proporcional a la distancia, ambos Beacons y sus lecturas mantienen una correlación lineal, ya sea positiva o negativa, ya que ambos aumentan o disminuyen en un mismo valor según la distancia obtenida, siempre y cuando esta sea pequeña o ambas señales se propaguen en la misma dirección espacial en un vecindario determinado. Esto se conoce tradicionalmente como correlación espacial de las señales.

Como la técnica elegida es PCA, a continuación, se procede a describir como funcionara específicamente para el problema de Fingerprint. PCA, busca la proyección sobre la cual los datos queden mejor representados en términos de mínimos cuadrados. Esta convierte un conjunto de variables posiblemente correlacionadas en un conjunto de variables sin correlación lineal, las cuales se denominan componentes principales. PCA construye una transformación lineal que escoge un nuevo sistema de coordenadas para el conjunto original de los datos, escogiendo la varianza mayor de los datos como primera componente en el nuevo sistema de referencia, la segunda varianza mas grande como segunda componente y así sucesivamente. Para ello se requiere la matriz de covarianzas, mediante la cual es posible encontrar los vectores propios de la misma, que funcionan como base para las nuevas coordenadas a través de la transformación lineal y así reducir la dimensionalidad.

PCA resuelve el problema de la alta dimensionalidad del radiomap construido, combinando los *features* mediante una transformación lineal en un espacio no correlacionado, es decir, un espacio ortogonal de vectores (vectores propios), utilizando la matriz de covarianza de los datos de entrenamiento de tamaño $M \times M$. Luego, este radiomap de señales RSSI es proyectado en el espacio no correlacionado en la dirección de la varianza más alta. La selección respectiva de componentes principales se basa en las varianzas más grandes, es decir, según los valores propios más grandes (proyección de la transformación). Por lo anterior, los valores propios representan la información de las componentes principales.

El espacio característico es generado mediante un conjunto de M lecturas RSSI por posición o punto de referencia, lo cual corresponde a un vector x_i de tamaño $1 \times M$ como vector fila, donde M es menor a N , y con N el número de ejemplos de entrenamiento.

Luego, el espacio característico posee una media:

$$\bar{x} = \sum_{i=1}^N \frac{x_i}{N} \quad (4.15)$$

Por ende, la matriz de covarianzas puede ser descrita como:

$$C_r = \sum_{i=1}^N (x_i - \bar{x})^T (x_i - \bar{x}) = X^T X \quad (4.16)$$

Cabe destacar que la matriz de covarianzas es simétrica y de tamaño $M \times M$, por lo cual posee M valores propios.

Sean los valores propios $\{\lambda_1, \lambda_2, \dots, \lambda_M\}$ ordenados en orden descendiente, con sus correspondientes vectores propios normalizados $\{V_1, \dots, V_M\}$. Según la fórmula de valores y vectores propios para matrices debe cumplirse entonces:

$$C_r V_i = \lambda_i V_i \quad (4.17)$$

$$\lambda_1 \geq \lambda_2 \geq \dots \geq \lambda_M \quad (4.18)$$

Como se menciona anteriormente, los vectores propios $\{V_1, \dots, V_M\}$ son no correlacionados y ortonormales, por lo que forman un espacio característico.

Con todo lo anterior, es posible proyectar el radiomap en el nuevo espacio característico. Obviamente, si se proyecta sobre todas las componentes no existe un cambio, ya que se presenta la misma información, por lo cual es necesario determinar los valores propios más relevantes y que aportan mayor información(varianza). Para la selección de las componentes principales se toma en cuenta entonces la información contextual y el error que cada componente ha aportado. No existe manera automática de determinar el número adecuado de componentes principales, sin embargo en la experimentación es discutida una forma de poder llevar a cabo esta tarea.

Luego, el nuevo radiomap no correlacionado Z es calculado proyectando X en el espacio característico reducido W , el cual representa una combinación lineal de los vectores propios seleccionados, asociados a los valores propios más relevantes. Finalmente, la matriz Z del nuevo radiomap, con menor número de componentes está dada por:

$$W = V \cdot X \quad (4.19)$$

$$Z = X \cdot W \quad (4.20)$$

Luego, los clasificadores de máquinas de aprendizaje son entrenados mediante la utilización del radiomap Z , el cual tiene componentes reducidas y no dependientes, con lo cual el entrenamiento es mucho más rápido y se elimina la mayor parte de la información redundante al extraer la información más relevante.

Las principales ventajas de utilizar PCA entonces, corresponde a ([Salamah et al., 2016](#)):

1. PCA extrae la información importante del radiomap definido.
2. PCA reduce la matriz de datos multivariados sin perder mucha información, en donde los datos están descritos por muchas variables correlacionadas dependientes.

3. PCA reduce la complejidad mediante la disminución del número de componentes, logrando mejorar tiempos de entrenamiento y calculo online.

El siguiente paso entonces corresponde a entrenar los modelos definidos, que se explican en la parte experimental, los cuales son técnicas de máquinas de aprendizaje muy conocidos y que han presentado buenos resultados a lo largo de muchos problemas. Posteriormente, se seleccionan los mejores algoritmos, es decir, que presenten el mejor desempeño y luego son implementados. Una parte importante a definir en la etapa offline es la manera en que los modelos de machine learning funcionaran en el dispositivo, ya que hay dos versiones posibles de implementación, estas son utilizando un servidor o sin utilizar servidor.

Ambas estrategias deben entrenar los algoritmos en una maquina dedicada, por el consumo de recursos que son necesarios, sin embargo, una vez que los modelos están entrenados, deben ser utilizados en el dispositivo móvil. Para ello se puede exponer un servicio REST o API en donde el dispositivo móvil envía una nueva lectura RSSI al servidor, posterior a esto el servidor computa y determina la posición y retorna este valor al dispositivo móvil. Esta alternativa es ideal en entornos donde siempre existe conexión a internet y redes de telefonía e internet móvil. Esta alternativa es idónea, ya que es el servidor quien procesa los resultados, haciendo el posicionamiento más expedito, además de utilizar menos recursos en el teléfono y por lo mismo menos batería, ya que solo realiza conexiones de red de poco peso (solo transmite un vector de enteros).

En el caso de este trabajo, como se pretende utilizar en lugares donde no existe internet como mineras o estacionamientos subterráneos, la alternativa es desarrollar el entrenamiento en el servidor, pero luego portar los modelos a un dispositivo móvil. Esto es posible, debido a que la mayoría de los modelos desarrollados presentan ciertos componentes que pueden ser posteriormente replicados en otras máquinas sin necesidad de entrenar o re entrenar el modelo.

Todos los pasos anteriores describen a grandes rasgos el proceso completo de la etapa offline, desde la recolección de datos hasta el entrenamiento e implementación de los modelos en los dispositivos móviles.

4.2.2. Fase Online

Para la fase online se reconocen dos etapas principales, la primera es colectar un vector de señales RSSI en la posición actual del usuario, es decir, el vector de intensidad de la señal en donde cada componente representa la intensidad recibida por un Beacon o Access Point Bluetooth. La segunda etapa es proveer este vector de entrada a los algoritmos de aprendizaje supervisado. Para realizar esta tarea se deben tener en cuenta las normalizaciones realizadas y aplicar correctamente la transformación PCA del espacio característico antes de suministrar los datos a los algoritmos, ya que de otra manera las dimensiones serán incompatibles.

Para realizar esto, se debe proyectar el vector RSSI en el espacio característico de la ecuación 4.19, es decir W . Los valores proyectados en esta etapa son comparados con los modelos entrenados buscando de esta manera el valor estimado más cercano según el nuevo radiomap. Una vez que los algoritmos de clasificación proveen el resultado de la posición física, entonces la misma aplicación de la fase offline, es utilizada para mostrar en un mapa de tiempo real la localización actual de usuario.

La fase Online es muy simple, ya que solo se deben evaluar los nuevos valores en los algoritmos, por lo que es claro que la parte más importante de todo el proceso de fingerprint es la recolección adecuada de datos y el correcto entrenamiento de los algoritmos de *machine learning*. Esto es de suma importancia, ya que malos datos o malos parámetros de entrenamiento, eventualmente provocan problemas, es decir, reducir la precisión en la fase Online afectando severamente los resultados.

A continuación, la Figura 4.10 describe el procedimiento completo a desarrollar en este trabajo.

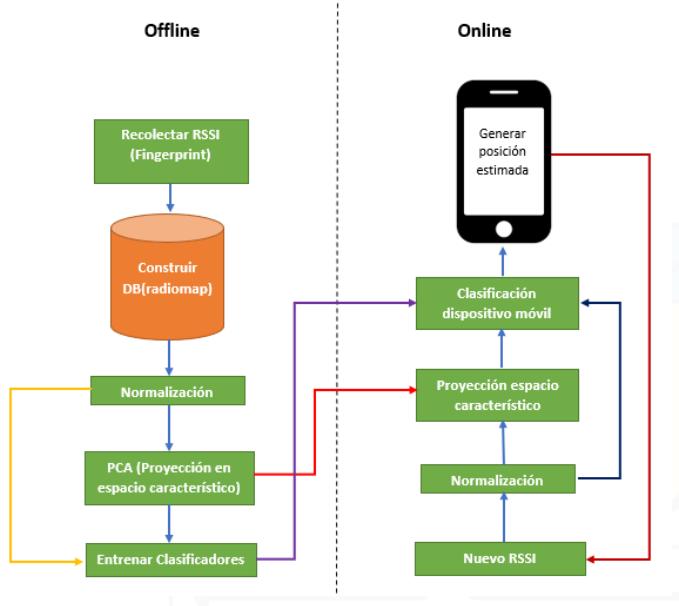


Figura 4.10: Framework desarrollado para resolver el problema de posicionamiento en interiores
(Fuente: Elaboración propia)

Como se observa en la Figura 4.10, los datos son normalizados previamente antes de ser utilizados. Además, la razón de la conexión entre la normalización y el entrenamiento de los clasificadores se debe a que se utilizaran los algoritmos de machine learning utilizando PCA y no utilizándolo, a modo de comparación en términos de tiempo y accuracy. Además, se ha omitido el ítem en donde los clasificadores ya entrenados son portados al dispositivo móvil. Finalmente, hay que notar que, en la fase online, el dispositivo móvil genera un nuevo vector de señales RSSI, luego se transforman los datos y finalmente se clasifican, lo que genera una posición estimada que se refleja en el dispositivo móvil. Este ciclo es constante, ya que la posición se actualiza continuamente según los parámetros definidos y frecuencia de actualización.

5 | Experimentación

5.1. Beacons y configuración

Los Beacons utilizados corresponden a Beacons de la marca Kontakt, y el número es 8, sin embargo, no son todos iguales. Esto ayuda a comprobar que tan efectivo es utilizar distintos modelos, ya que en entornos reales, por ejemplo en una mina, es sumamente importante contar con Beacons Bluetooth que se adapten al espacio físico en donde son dispuestos, por ejemplo si es necesario instalarlos en una oficina, los Beacons convencionales funcionan de buena manera, sin embargo en ambientes hostiles como lugares con polvo o de difícil acceso, es necesario contar con Beacons preparados para estas dificultades, los cuales tengan baterías de larga duración, sean anti polvo y agua, entre otras características. A continuación, se muestra el resumen de los Beacons utilizados:

1. **Beacon:** Beacon más popular y de menores prestaciones. Es el clásico Beacon vendido por la compañía Kontakt. Posee una batería con duración de hasta 48 meses y un rango de 70m. Se utilizan 4 unidades.
2. **Beacon Pro:** Beacon con más funciones de seguridad, ahorro de batería, sensores de luz y a prueba de agua. Hasta 60 meses de batería y rango de 70m. Se utilizan 2 unidades.
3. **Tough Beacon:** Hasta 24 meses de batería y rango de 70m. A prueba de polvo y agua. Especialmente diseñado para condiciones extremas.

Cabe destacar que las características básicas del *Hardware* de cada Beacon son las presentes en la comparativa realizada en la sección 4.1.3.



Figura 5.1: Diferentes tipos de Beacons utilizados en la experimentación
(Fuente: Elaboración propia)

Lo siguiente es definir la configuración de cada Beacon. Los parámetros más relevantes en este caso serían el *TX Power* y el *Beacon Interval*. En un despliegue real, lo más relevante es disminuir los costos, que en este caso están asociado a las baterías o más bien al cambio de estas a lo largo del tiempo. Estos dos parámetros son importantes para la localización indoor, ya que el TX Power define la potencia de salida de cada Beacon y el intervalo corresponde a cada cuanto un Beacon envía un mensaje a los dispositivos Bluetooth cercanos.

El valor de Tx Power afecta principalmente a tres factores claves, estos son el rango de la señal, es decir, la distancia máxima alcanzada; el segundo corresponde a la estabilidad de la señal y finalmente la batería, aunque este último factor es mayormente influenciado por el intervalo.

En términos simples, TX Power representa que tan poderosa es la señal transmitida por los Beacons en dBm y corresponde a un número dentro del rango 0-7, donde 0 es menos poderoso y 7 el más poderoso. Es claro que mientras más poderoso es la señal, mayor rango alcanza y más estable es la señal, sin embargo, el consumo de energía aumenta significativamente. Para determinar el parámetro TX power entonces es necesario tener en cuenta estos factores. Para el caso del posicionamiento indoor, es claro que el rango debe ser máximo, ya que de esta forma la señal de cada Beacon puede alcanzar a la gran parte de los usuarios de la zona. El otro punto significativo es establecer que tan seguido es necesario cambiar las baterías. Este punto es relevante para despliegues a gran escala y depende en gran medida de las organizaciones interesadas en desarrollar un sistema de este tipo. Finalmente, es importante considerar la interferencia subyacente debido al entorno, por ejemplo, objetos contundentes o múltiples personas, en este caso se requiere una señal más potente para cortar de esta forma la interferencia y pasar a través de esta. Por los motivos declarados anteriormente se decide utilizar un TX Power igual a 7, o equivalentemente 4 dBm.

Con respecto al intervalo utilizado, es un parámetro de mucho mayor relevancia que el TX Power para el tipo de problema a resolver. Esto se debe principalmente a que el intervalo corresponde a que tan frecuente es transmitido un paquete desde un Beacon a un dispositivo móvil. Considerando que el posicionamiento debe ser lo más cercano a tiempo real, la información debe ser suministrada lo más rápidamente posible. El intervalo es medido en milisegundos, y como es esperable, mientras más pequeño el intervalo, más recurrente es la comunicación, pero mayor es el consumo de batería. A pesar de que el intervalo sea muy pequeño, muchas veces los sistemas operativos no son capaces de recibir tan rápidamente estas señales, debido a limitaciones de seguridad u otras configuraciones del fabricante como es el caso de Apple. En Android estas limitaciones no existen y por lo tanto puede recibir todas las señales, pero esto afecta de forma significativa la batería del teléfono. Por otra parte, mientras menor es el intervalo, mayor es la estabilidad de la señal, además el intervalo afecta significativamente a el posicionamiento en interiores, debido a que mientras más rápido se mueve un usuario, si el intervalo es demasiado alto, su posición sufrirá saltos o cortes que no muestran realmente la ruta de desplazamiento. Si el intervalo es pequeño, puede detectar más fácilmente la posición del usuario en cada momento con precisión de unos pocos centímetros. La Figura 5.2 ejemplifica esta situación:

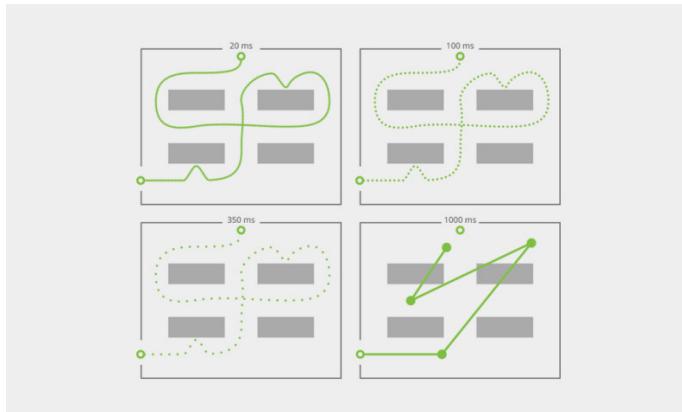


Figura 5.2: Posicionamiento indoor con diferentes intervalos de aviso en milisegundos

(Fuente: [Agnieszka Gasiorek \(2014\)](#))

Por lo anterior, es claro que los parámetros seleccionados deben ser el mayor valor posible para el TX Power correspondiente a 7, y un valor del intervalo igual a 100ms. Para configurar estos parámetros, Kontakt dispone de una aplicación para dispositivos Android. La Figura 5.3 y 5.4 muestran como han sido configurados los Beacons. Esta configuración es idéntica para todos los Beacons utilizados.

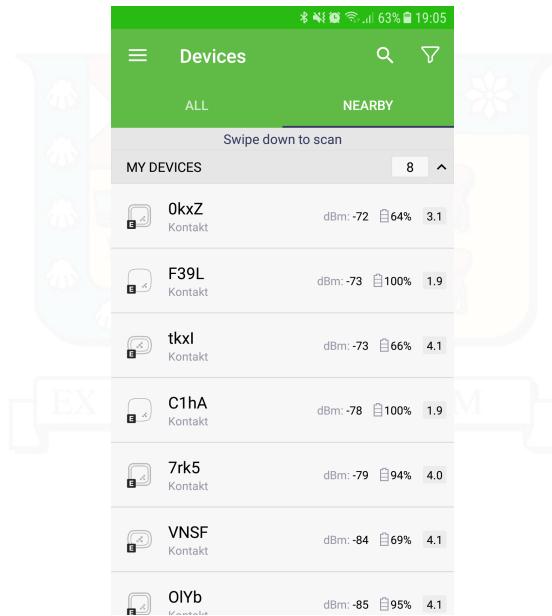


Figura 5.3: Listado de los Beacons encontrados para su posterior configuración

(Fuente: Elaboración Propia)

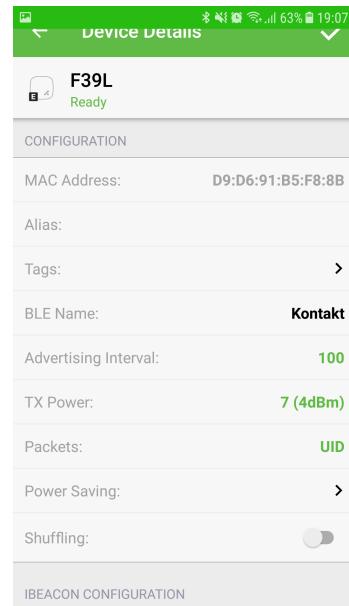


Figura 5.4: Configuración que ejemplifica los valores definidos para TX Power y el intervalo de transmisión

(Fuente: Elaboración Propia)

Además, los protocolos utilizados son UID de Eddystone por su flexibilidad y transversalidad para los múltiples dispositivos móviles existentes en el mercado.

5.2. Lugar de experimentación

Para realizar los experimentos, se decide utilizar un lugar en donde las condiciones sean adversas y cambiantes, debido al alto tránsito de objetos y además no se presenten señales GPS. Por lo anterior, se decide utilizar el estacionamiento subterráneo de la universidad Técnica Federico Santa María, Campus San Joaquín, ubicado en Vicuña Mackenna 3939. Las dimensiones constan de 144,75m de largo y 36m de ancho. La Figura 5.5 muestra el plano del estacionamiento a escala.

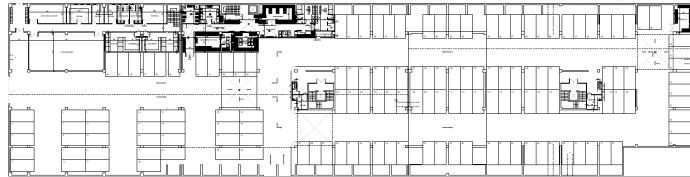


Figura 5.5: Plano del estacionamiento subterráneo de la Universidad Federico Santa María

(Fuente: Elaboración Propia)

Este lugar es ideal para realizar pruebas, ya que presenta interferencia debido a los vehículos en tránsito y la disposición de los objetos es cambiante. Además, se pueden obtener mediciones con ruido producto de la reflexión de las ondas, lo que pone a prueba a los algoritmos de clasificación para detectar estos patrones.



Figura 5.6: Fotografía del espacio de experimentación real, correspondiente al estacionamiento subterráneo

(Fuente: Elaboración Propia)

5.3. Software Utilizado

5.3.1. Descripción de la aplicación

La aplicación desarrollada es construida para el sistema operativo Android. Para su desarrollo se utiliza Android Studio¹, es IDE oficial para el desarrollo de aplicaciones nativas. El número de compilación corresponde a 3,0,1. La selección de este sistema operativo es debido a su fácil acceso y prácticamente ningún costo debido a licencias, ya que es basado en Linux por lo que su desarrollo es gratuito, al igual que la

¹<https://developer.android.com/studio/index.html>

distribución de aplicaciones, por lo que es mucho más efectivo para el desarrollo de una aplicación de prueba bajo un entorno controlado.

Los requisitos básicos de la aplicación construida son los siguientes:

- Mostrar el plano del lugar de experimentación, el cual funciona como base para posicionar los demás elementos.
- Permitir la adición de nuevos dispositivos Beacons con su respectivo marcador. Para ello debe detectar automáticamente el Beacon más cercano y sugerir este para su inserción en el mapa.
- Permitir la captura de datos, es decir, los nuevos Fingerprints, en donde el usuario es quien provee la posición exacta y la aplicación mide los valores RSSI durante un periodo de tiempo, para posteriormente almacenarlos en la base de datos.
- Modificar los valores de intervalo y el número de mediciones en cada punto, el cual puede también definirse en periodo de tiempo.
- Tener una base de datos *SQLite*, la cual permite la persistencia de los datos. También se debe poder transformar esta base de datos a un archivo CSV.
- Para la etapa online, debe permitir seleccionar el algoritmo a utilizar y mostrar en tiempo real la posición del usuario según el algoritmo usado. Por otra parte, debe guardar estos resultados en un archivo de texto para su posterior análisis.

La aplicación construida consta de 3 secciones principales las cuales ayudan a todas las etapas del modelo Fingerprint, estas son **Offline**, **Online** y **Configuración**. Estas son descritas a medida que se desarrolla el proceso de la experimentación.

5.3.2. scikit-learn

Para la elaboración de modelos de clasificación, una de las librerías más utilizadas es scikit-learn o también conocida como SKlearn [Pedregosa et al. \(2011\)](#), la cual es una completa API para elaborar y realizar maquetas de machine learning. Su Core está basado en Python y para su uso se utiliza el siguiente entorno de desarrollo:

- Anaconda 1.6.2
- Jupyter Notebook 5.0.0
- scikit-learn 0.18.2
- Python 3.6.1

Las principales ventajas de utilizar SKlearn son obtener modelos simples y eficientes, que están muy avanzados y que presentan una comunidad dedicada a su correcto funcionamiento. Además está construida sobre 3 de las librerías más utilizadas en Python en el contexto de desarrollo científico, como son NumPy, SciPy y Matplotlib. Por otra parte, su licencia es de código abierto, por lo que cualquiera puede utilizar y modificar el código sin restricciones.

A pesar de que en este caso solo se utilizará para los algoritmos de clasificación, PCA y gráficos, también presenta herramientas que agilizan el desarrollo como por ejemplo lectura de archivos, funciones de normalización establecidas, funciones de puntuación o accuracy, pre procesamiento de datos, entre otras. También sirve como herramienta de regresión, minería de datos y análisis de datos.

5.3.3. Tensorflow

A pesar de que SKlearn presenta tantas ventajas para el desarrollo ágil de modelos de máquinas de aprendizaje, no es tan bueno en redes neuronales, y no implementa la mayor parte de los algoritmos modernos

en este campo. Esto se debe principalmente a que todo el Core de SKlearn es Python, y se vuelve sumamente lento para entrenar este tipo de redes, sobre todo redes profundas. Para resolver este problema, se decide utilizar Tensorflow [Abadi et al. \(2015\)](#), un completo Framework de desarrollo de Google, que con respecto a SKlearn es mucho menos intuitivo y de fácil uso al usuario final, ya que su programación es de mucho más bajo nivel y requiere conocimiento avanzado de los algoritmos que se quieren implementar. Muchas veces se describe como las piezas fundamentales para la creación de algoritmos de máquinas de aprendizaje, como si fueran Legos, no así SKlearn, el cual ya está todo listo, solo debe ser utilizados y configurar los parámetros de los algoritmos.

Tensorflow es una biblioteca de código abierto y su principal ventaja es que a pesar de que la parte superior está basada en Python, esta se comunica internamente a código compilado en C++, lo cual lo vuelve sumamente eficiente. Además, por la forma en que Tensorflow representa los datos, es decir, tensores o arreglos multidimensionales, y las operaciones referentes a grafos sin estado que transforman estos tensores, lo vuelve increíblemente rápido para entrenar y desplegar, incluso presenta API para dispositivos móviles en Java, C++ y GO, en donde solo se porta el grafo construido y se puede realizar inferencia en equipos de muy pocos recursos, lo que lo vuelve una alternativa ideal para desarrollar redes neuronales profundas, utilizando pocos recursos en el sistema operativo Android. A pesar de que presenta operaciones de bajo nivel, igualmente tiene disponible la opción de API de alto nivel, como por ejemplo implementar redes convoluciones o redes profundas en un par de líneas de código Python, conocidos como estimadores de alto nivel.

Finalmente, Tensorflow basa todo su procesamiento en el grafo computacional mencionado anteriormente, el cual consiste en una serie de operaciones de tensores dispuestos en un grafo de nodos. Luego de que se arma este grafo de operaciones, se puede entrenar mediante las operaciones especiales incluidas en Tensorflow, llamados optimizadores, como lo son gradiente descendente. El paso final es utilizar este grafo computacional para probar los modelos y utilizarlos en despliegues reales.

5.4. Recolección de Fingerprints

Con todo lo anterior en cuenta, es necesario definir qué tipo de dispositivo móvil y sistema operativo se utilizará para colectar los datos en cuestión, que finalmente se transformaran en Fingerprint dando lugar al radiomap. En este caso, el dispositivo a utilizar es un dispositivo Samsung Galaxy J7 Prime, que dentro de sus prestaciones posee una CPU Octa-core 1.6 GHz Cortex-A53, una GPU Mali-T830 MP1, 3GB de memoria ram interna y el tipo de Bluetooth corresponde a 4.1 LE. Con respecto a su sistema operativo, es Android 7.0 Nougat el cual no presenta limitaciones en la lectura de las señales Bluetooth por lo que puede recepcionar las señales tan rápido como estas son emitidas.

Definido esto, lo siguiente establecer la forma en que deben ser posicionados los Beacons y además cómo funciona la recolección de Fingerprints como tal. Para el posicionamiento de los Beacons, se decide utilizar una estrategia de grilla, esto es, ubicarlos de manera equidistante en las esquinas del plano. Además, para este trabajo se debe considerar que el plano completo no fue inspeccionado debido a que la densidad de los Beacons sería muy baja para lograr óptimos resultados, por lo que finalmente se decide utilizar los 8 beacons en un área reducida del estacionamiento y ubicar cada Beacon a una distancia de 16 metros a sus vecinos adyacentes formando esta especie de grilla. Finalmente, el área efectiva a utilizar se calcula según las dimensiones de 16 metros de ancho por 44 metros de largo, con lo que se obtiene un área de $704m^2$. Cabe destacar que un par de Beacons fueron ubicados respecto a sus vecinos a una distancia de 12 metros y no 16, debido a la disposición del estacionamiento. Lo anterior define un cuadrilátero de esquinas $\{(0, 6), (0, 22), (44, 22), (44, 6)\}$, Notar que el valor de Y comienza en 6 y no en 0, debido a dificultades en el lugar de experimentación.

Con respecto al punto de recolección de Fingerprints, se desarrolla una aplicación para el sistema operativo Android, y se decide utilizar una grilla para los puntos de medición o referencia de 4 metros por 4 metros, es decir, cada punto de medición está en medio de cada celda de esta grilla, consiguiendo de esta manera un total de 44 puntos de referencia. La Figura 5.7 muestra esta situación.

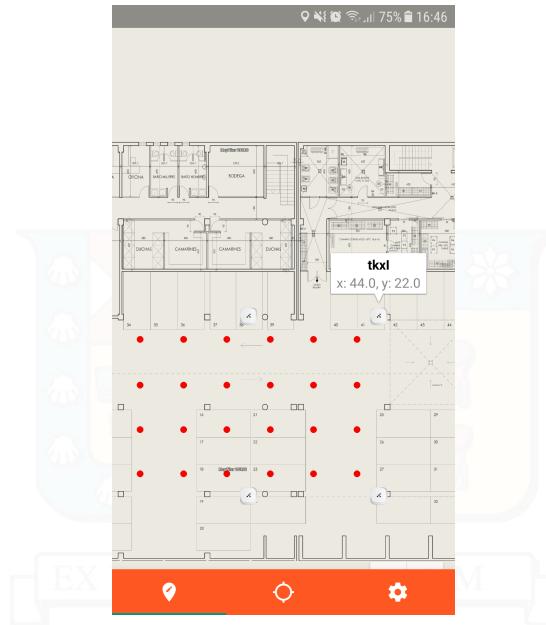


Figura 5.7: Disposición de Beacons y puntos de referencia dentro de la grilla, utilizando la aplicación Android
(Fuente: Elaboración Propia)

La aplicación consta de una sección offline, en donde se pueden incorporar nuevos Beacons a la base de datos *sqlite*(base de datos por defecto en dispositivos Android), además de poder realizar la captura de datos o Fingerprints, definiendo la ubicación actual según algún patrón o posición exacta. Otro punto importante a considerar es el número de Fingerprints a obtener en cada punto de referencia. Considerando el intervalo de emisión de cada Beacon, es necesario establecer un tiempo alto para notar cambios en el lugar de experimentación, por lo que se define una ventana de tiempo de 5 minutos en cada punto de referencia, o equivalentemente 300 segundos, con lo cual se obtienen 3000 mediciones por posición. Sin embargo, para obtener mejores resultados y en vista y considerando que las máquinas de aprendizaje aprenden de mucho mejor manera con datos variados, se decide inspeccionar y recolectar datos a través de diferentes días, con el objetivo de que sea más fácil para los algoritmos encontrar patrones eventualmente. Finalmente se seleccionan 150 mediciones por punto de la grilla, para tener menor información repetida y no sobre muestrear la base de datos. En este caso, para la selección de los 150 valores, se utiliza un selector aleatorio, de cada día utilizado, en este caso 3 días con 50 mediciones de cada día, es decir, un dato cada 6 segundos, con el objetivo de obtener datos lo más representativos posibles.

Una vez recolectados los datos, se obtiene una base de datos *SQLite* con Fingerprints, la cual presenta 6600 registros, en donde los dos primeros campos representan la posición de X y la posición de Y respectivamente, y los siguientes 8 campos corresponden a los Beacons detectados con su respectiva intensidad de la señal o RSSI. Por lo anterior, las clases definidas para X son $X \in [2, 6, 10, 14, 18, 22, 26, 30, 34, 38, 42]$ y para $Y \in [8, 12, 16, 20]$ con un total de 11 clases para X y 4 clases para Y. Recordando que estas clases representan el centro de la grilla con celdas cuadradas de 4 metros, es decir el punto medio de estas, por lo que para X, la primera clase se define según el punto inicial, el cual es cero, además el punto medio de la primera celda es la mitad de la dimensión total, es decir, 2 metros que representa la primera clase y así se construyen las siguientes clases, adicionando 4 metros(cambio de celda). Para Y se debe tener en consideración que la primera clase es 8, esto se debe principalmente a que los valores de Y no comienzan precisamente en cero, debido a dificultades en la posición de objetos dentro del estacionamiento, por lo que se decide iniciar en $Y = 6$, con lo que el primer punto es este valor inicial más la mitad de la grilla(2), es decir 8. Un punto importante a definir es que valor asignar a los Beacons que no han sido detectados, en este caso y como el valor RSSI no puede ser mayor a cero, se decide asignar un valor de 100 dBm como un identificador de que no se ha detectado señal.

Con esta base de datos es posible construir otros tipos de archivos, que resuman todos los Fingerprints

detectados. Esto se realiza ya que para los algoritmos de machine learning existen muchas más librerías dedicadas a leer datos en formatos más convencionales como TXT o CSV. Por lo mismo, en la aplicación se define una sección de configuración para transformar y almacenar esta base de datos en formato CSV o *comma-separated values*. La Figura 5.8 muestra esta pestaña de configuración, incluyendo el número de mediciones y las opciones de exportar la base de datos, transformarla a CSV o eliminarla.

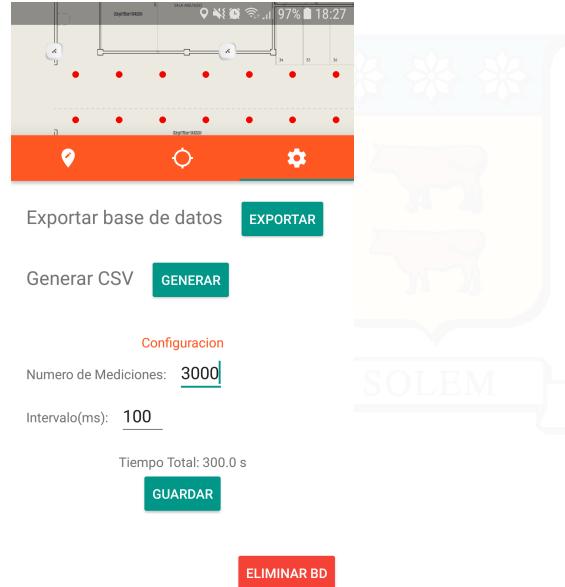


Figura 5.8: Pestaña de configuración de la aplicación Android, donde se puede configurar el tiempo y exportar la base de datos a CSV

(Fuente: Elaboración Propia)

Un ejemplo del archivo CSV obtenido se presenta en la Figura 5.9

	A	B	C	D	E
1	X,"Y","C1hA","0kxZ","tvMX","Olyb","7rk5","F39L","VNSF","tkxl"				
2	2.0,"8.0","-66","-92","-84","-84","-92","-93","-98","-96"				
3	2.0,"8.0","-66","-94","-84","-84","-94","-93","-98","-96"				
4	2.0,"8.0","-66","-94","-84","-84","-94","-93","-98","-100"				
5	2.0,"8.0","-66","-94","-84","-84","-94","-80","-98","-100"				
6	2.0,"8.0","-66","-94","-84","-84","-94","-74","-98","-100"				
7	2.0,"8.0","-66","-94","-84","-84","-92","-74","-98","-100"				
8	2.0,"8.0","-66","-97","-81","-84","-91","-74","-98","-97"				
9	2.0,"8.0","-70","-97","-84","-84","-91","-74","-96","-97"				
10	2.0,"8.0","-70","-97","-84","-86","-91","-74","-96","-97"				

Figura 5.9: Segmento del archivo CSV obtenido a partir de la base de datos. Los primeros dos campos corresponden a las coordenadas y los siguientes, los id únicos de cada Beacon utilizado, con su respectivo valor RSSI en dBm. Cada registro corresponde a un Fingerprint.

(Fuente: Elaboración Propia)

Todos estos pasos de recolección son necesarios para los posteriores análisis y entrenamiento de los algoritmos de máquinas de aprendizaje. Para entrenar los clasificadores, se considera entonces cada Beacon en un registro como un feature, es decir, un atributo que funcionara como predictora para la clasificación. Las etiquetas de cada registro pueden corresponder tanto a X como a Y, entonces debe entrenarse en ambos casos, por lo que cada clasificador debe presentar 4 casos, esto es X e Y con PCA y X e Y solo con normalización de los datos.

5.5. Visualización de los datos obtenidos

En máquinas de aprendizaje, siempre es bueno analizar los datos para determinar patrones o el comportamiento general de los datos. Claramente no es posible visualizar todas las características de una vez, ya que, al ser más de tres dimensiones, es complejo para la visualización humana. Mientras los datos en dos o tres dimensiones pueden ser graficados e interpretados, los datasets de altas dimensiones no cuentan con la misma suerte, y ellos representan la mayor parte de los problemas existentes. El fin de determinar la estructura de los datos es reconocer por ejemplo algún comportamiento extraño o ruido, el cual posteriormente puede ser removido en el caso de ser necesario. Para permitir la visualización de la estructura del dataset, la dimensión debe ser reducida de alguna manera, para ser representada en un plano o el espacio cartesiano, algo que es muy intuitivo a la visualización humana.

Para sobrellevar esto, la manera más simple es tomar una proyección aleatoria de los datos, es decir, seleccionar los features más representativos según alguna especie de peso o ranking. Esto permite algún grado de visualización de la estructura de los datos, pero se pierde información que muchas veces puede ser relevante. En este tipo de proyecciones, la estructura más interesante de los datos se pierde debido a la falta de información, por lo que no se representa el problema completo.

Además, existen muchos algoritmos tanto supervisados como no supervisados, que son lineales y permiten la reducción de la dimensionalidad, como son **PCA**, análisis de componentes independientes, análisis de discriminante lineal(**LDA**), entre otros. En este trabajo PCA es utilizado para reducir la dimensionalidad y evitar la correlación lineal entre los features de los datos. Estos algoritmos definen formas de realizar proyecciones lineales interesantes de los datos. Aunque estos métodos son frecuentemente utilizados, es importante además definir otro tipo de algoritmos que ayuden a establecer si entre todas las variables existen estructuras no lineales.

Para conseguir esto, a continuación se utilizan técnicas de **Manifold Learning**, las cuales pueden generalizar los frameworks lineales como PCA para ser sensibles a datos altamente no lineales. Un punto importante a destacar es que la mayor cantidad de algoritmos de este tipo son no supervisados, por lo que pueden encontrar las clases automáticamente y separarlas a partir de los mismos datos, sin necesidad de suministrar las etiquetas de cada fingerprint.

Los algoritmos a comparar no serán abordados en detalle porque no es el objetivo de este trabajo, solo se analizarán sus resultados respectivos. Los algoritmos utilizados son Locally Linear Embedding de tipo standar, Itsa, hessian y modificado. Además se utilizan otros cuatro métodos correspondientes a Isomap, *Multi-dimensional Scaling*, *Spectral Embedding* y finalmente *t-distributed Stochastic Neighbor Embedding(t-SNE)*. Para utilizar estos métodos en primer lugar se normalizan los datos mediante SKlearn, dejándolos con media igual a cero y varianza unitaria, es decir, datos normalmente distribuidos como se define a continuación:

$$x' = \frac{x - \bar{x}}{\sigma} \quad (5.1)$$

Una vez estandarizados los datos, se procede a evaluar los algoritmos en ellos. Cabe destacar que para realizar la evaluación se utilizan las clases de la etiqueta *X*, a pesar de que puede usarse *Y* igualmente, por lo mismo el número de clases corresponde a 11. La Figura 5.10 resume los resultados obtenidos debido a aplicar los algoritmos anteriormente mencionados para una proyección en dos componentes, es decir en 2-D, con 10 vecinos, donde los vecinos representan un área o vecindario similar al que utiliza K-NN.

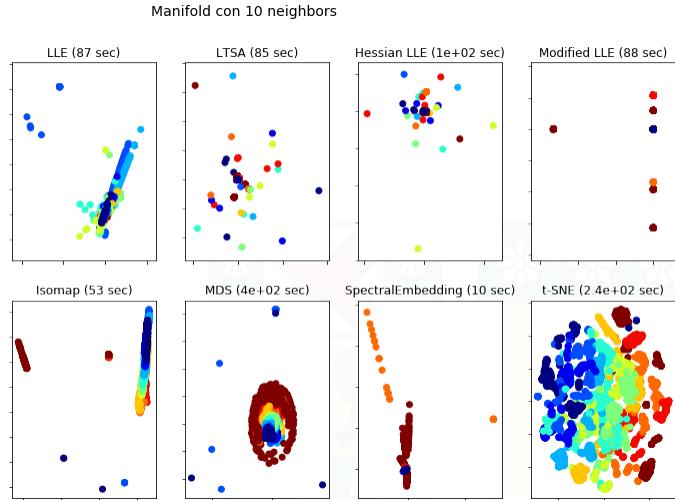


Figura 5.10: Comparación de los diferentes algoritmos Manifold para la reducción de la dimensionalidad con sus respectivos tiempos de ejecución.

(Fuente: Elaboración Propia)

Como se aprecia en la figura, la mayor parte de los algoritmos no logran determinar correctamente la estructura de los datos ni mucho menos separar las clases. Los mejores resultados obtenidos en este caso corresponden a Isomap, MDS y t-SNE. Isomap es una extensión de MDS y Kernel PCA, y su objetivo es mantener las distancias geodésicas entre todos los puntos. Por su parte MDS busca una representación en la cual las distancias son respetadas según el espacio original. Finalmente, t-SNE convierte la afinidad de los puntos en probabilidades, lo cual lo convierte en uno de los algoritmos más utilizados, ya que es muy sensible a las estructuras locales en los datos, además reduce la tendencia de los datos a agruparse, revelando su estructura de múltiples escalas en un solo mapa. Estos Manifolds muestran que los datos presentan una estructura no lineal, debido a los resultados obtenidos principalmente por Isomap y MDS, ya que ellos mantienen las distancias entre los puntos, por lo que si se lleva al espacio 8-dimensional, MDS muestra una clara separación entre clases anidadas, lo cual en un espacio de mayor dimensión, se puede traducir en clases ubicadas cada una en la superficie de un hiperboloide o un hiperplano, lo cual muestra la no linealidad entre los features, sin embargo igualmente se muestran tendencias lineales, pero esto se debe entender como una correlación lineal local. t-SNE no logra separar las clases de manera correcta, lo cual se puede interpretar como que los datos no son completamente no lineales, y muestran rasgos de linealidad, por lo que PCA se puede utilizar teniendo en cuenta esta aproximación, ya que la correlación no lineal entre las variables no es fuerte. Con lo anterior, es claro que los valores RSSI ciertamente fluctúan mucho, debido a la gran cantidad de outliers, lo cual es un problema en términos de algoritmos de máquinas de aprendizaje, por situaciones que serán evaluadas posteriormente.

5.6. Análisis PCA y LDA

A pesar de que LDA ya fue descartado previamente debido a que no provee lo que se busca en este trabajo, más precisamente, disminuir o eliminar la correlación entre puntos adyacentes en la grilla, igualmente se analiza su comportamiento en el dataset obtenido para tener una mejor comprensión de que tanta linealidad presentan los datos, ya que este es un método muy efectivo cuando los datos son etiquetados como en este caso.

Es necesario escalar los datos nuevamente, ya que PCA puede ser susceptible a la escala de los datos y provocar resultados arbitrarios, por lo que se centran los datos, con media en cero y varianza unitaria. La resta de la media es necesaria para asegurar que la primera componente principal describe la dirección de máxima varianza, por lo que una media cero es necesaria para asegurar una base en el nuevo subespacio que

minimice el error cuadrático medio. Luego de estandarizar los datos, se procede a aplicar PCA y LDA en ellos, de donde se obtiene la Figura 5.11.

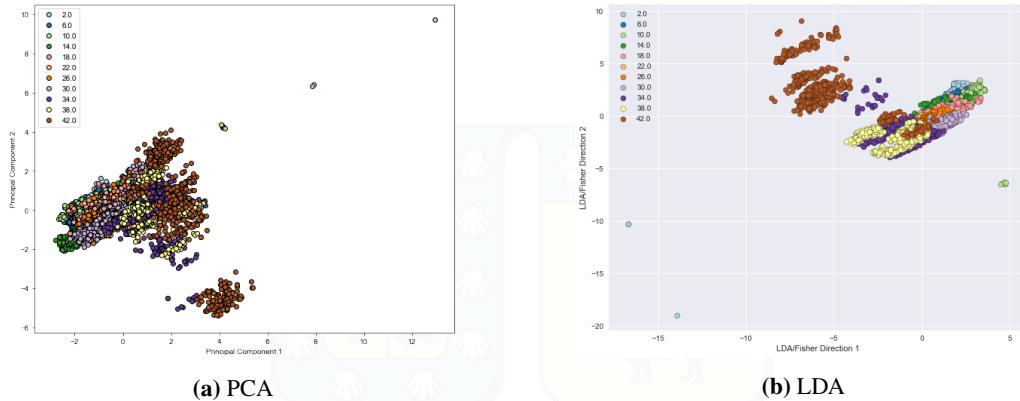


Figura 5.11: Comparativa entre los algoritmos PCA y LDA aplicado a los datos

(Fuente: Elaboración Propia)

Como se aprecia en la figura, para PCA se proyecta en las primeras dos componentes principales, lo cual no obtiene buenos resultados, esto puede deberse a muchos factores presentes en los datos, ya que como se mencionó anteriormente, los datos no son completamente lineales en el espacio multidimensional, por lo que PCA puede presentar problemas. Además, la correlación de los datos es lineal pero solo localmente, en la imagen global, la correlación debe ser de un orden superior (no lineal), por lo que PCA no puede obtener buenos resultados. También es necesario destacar que PCA no está optimizado para separación de las clases, y no es precisamente lo que se busca al utilizarlo, la idea principal es reducir la dimensionalidad y evitar la correlación espacial presente.

Con respecto a LDA, este si presenta buenos resultados, logrando separación entre las clases en gran parte. Esto se debe a que el objetivo de LDA es encontrar una combinación lineal de los features que separe las clases. Además, funciona como un clasificador, esto explica los mejores resultados en comparación con PCA. Con lo anterior, es claro la tendencia lineal local de los datos, por lo que al aplicar PCA los algoritmos de clasificación deberían presentar resultados positivos debido a que solo se omite información no relevante.

5.7. Entrenamiento de clasificadores

Para el entrenamiento de los clasificadores, se debe tener en cuenta como se menciona anteriormente, que para la gran mayoría de los algoritmos es necesario una normalización previa. Esto es particularmente importante en problemas en donde cada atributo puede diferir mucho en orden de magnitud con otras variables. En este caso, no es ciertamente un requisito, ya que los atributos poseen valores de la misma magnitud, porque todas reflejan valores en dBm en un mismo rango, sin embargo, igualmente es una buena práctica para algoritmos basados en distancia como KNN o SVM, sobre todo por la inclusión del valor 100dbm incorporado para denotar la ausencia de un dispositivo Beacon.

Los clasificadores seleccionados en para este trabajo son Naive Bayes, SVM con Kernel de tipo Radial Basis Function(**RBF**), SVM con Kernel lineal, K-NN, Arboles de decisión, Random Forest, Adaboost, Neural network y QDA. Los parámetros de cada algoritmo son configurados según una búsqueda en grilla, por lo que es necesario correr cada algoritmo múltiples veces en busca de los mejores parámetros. Para realizar las pruebas, la forma clásica de realizar machine learning indica tener un training set y un test set, sin embargo, en este caso no existe test set, por lo que una opción muy utilizada es separar el conjunto completo de ejemplos en entrenamiento y pruebas, seleccionando un porcentaje del total como test set. Esto funciona muy bien cuando el número de ejemplos es suficientemente grande, lo cual no ocurre en este caso, por lo que

si se quitan datos al training set, es posible que los clasificadores obtengan resultados muy por debajo de cómo lo harían con la totalidad de los datos.

Para resolver el problema anteriormente descrito, existe una técnica denominada cross-validation, la cual es muy utilizada y conocida en los campos de inteligencia artificial. El problema principal es que los parámetros de cada modelo se ajustan a los datos de entrenamiento lo mejor que pueden, pero al tomar una muestra independiente como datos de prueba desde el mismo training set, generalmente el modelo no se ajusta igual de bien a estos datos de prueba que a los datos de entrenamiento, lo cual se conoce como sobreajuste u *overfitting* y ocurre habitualmente en training set pequeños o cuando el número de parámetros del modelo es muy grande. Entonces la técnica de cross-validation ayuda a prevenir estas situaciones, ya que, a partir de un número determinado de iteraciones, en cada una de ellas selecciona un conjunto de pruebas de tamaño determinado desde el conjunto total, y al final de todas las iteraciones realiza un promedio de los errores, con lo cual se puede estimar el mejor test score.

En este caso, se decide utilizar cross-validation modificado, el cual utiliza 20 % de los datos totales como test set, es decir, lo que habitualmente se conoce como 5-fold cross-validation ya que son 5 iteraciones generales en donde el conjunto de entrenamiento se divide en 5 subconjuntos. La modificación es que, en cada iteración general, además se realizan 100 iteraciones adicionales repitiendo el test set, por lo que en total se realiza un total de 500 iteraciones. Esto se debe a que de esta forma se obtienen curvas más suaves respecto a la media del test y train set. Para obtener el mejor valor en cada caso, se promedian las 100 iteraciones de la primera iteración global y se obtiene un valor, esto se realiza para cada una de las 5 iteraciones y finalmente se promedian los 5 valores (uno por cada iteración global) para obtener el mejor valor del train y test set.

Un punto a considerar es como se mencionó anteriormente, las redes neuronales no siguen este procedimiento debido a que sklearn no posee este tipo de técnicas, por lo que se usa Tensorflow. Tensorflow tiene sus propios métodos de validación y los parámetros en este caso son configurados mediante ensayo y error, debido a que esta forma presenta buenos resultados en redes neuronales profundas debido a su complejidad.

Los gráficos obtenidos se muestran conjuntamente en la Figura 5.12.

Se debe señalar que los gráficos son para la clasificación sobre la coordenada X, sin embargo para Y los resultados son similares, producto de que los datos son esencialmente los mismos, lo único que cambia en el entrenamiento son las clases, así que en términos generales los resultados no deben variar mucho para los algoritmos que realicen una buena clasificación.

Lo que muestran las curvas son los valores medios obtenidos en cada una de las divisiones o *folds*, a partir de las 100 iteraciones. La parte sombreada de cada curva representa la desviación estándar, es decir, que tanto varían los resultados obtenidos respecto a la media, lo cual sirve para determinar si las pruebas son precisas y no solo depende de la partición realizada en el train set y test set. Por otra parte, para obtener el valor del accuracy se utiliza los *scoring* definidos por defecto en cada clasificador, que en su mayoría corresponde a la similaridad de Jaccard o índice de Jaccard, el compara la similaridad de dos conjuntos y está definida como el tamaño de la intersección de los conjuntos dividida por el tamaño de la unión.

$$J(A, B) = \frac{|A \cap B|}{|A \cup B|} = \frac{|A \cap B|}{|A| + |B| - |A \cap B|} \quad (5.2)$$

Se debe notar que el gráfico de entrenamiento y validación de las redes neuronales profundas no se encuentra presente en la Figura 5.12, esto es debido a como se menciona anteriormente, los clasificadores convencionales son implementados en la librería SKlearn, mientras que la red neuronal debe ser procesada en un framework desarrollado exclusivamente para este tipo de redes, por su complejidad computacional y obtener resultados optimizados. Para ello se utiliza Tensorflow y la red utilizada es una red neuronal profunda con dos capas ocultas, la primera de ellas tiene 256 neuronas o nodos, mientras que la segunda capa posee 64 neuronas. Además, cabe mencionar que para realizar el entrenamiento de redes neuronales habitualmente no se utiliza cross-validation, ya que cada iteración supone mucho tiempo, por lo que el entrenamiento total tiene un alto costo computacional, y para grandes bases de datos se utilizan conjuntamente procesadores y tarjetas gráficas para un mejor rendimiento. Por lo anterior, y en vista que cross-validation exige entrenar el

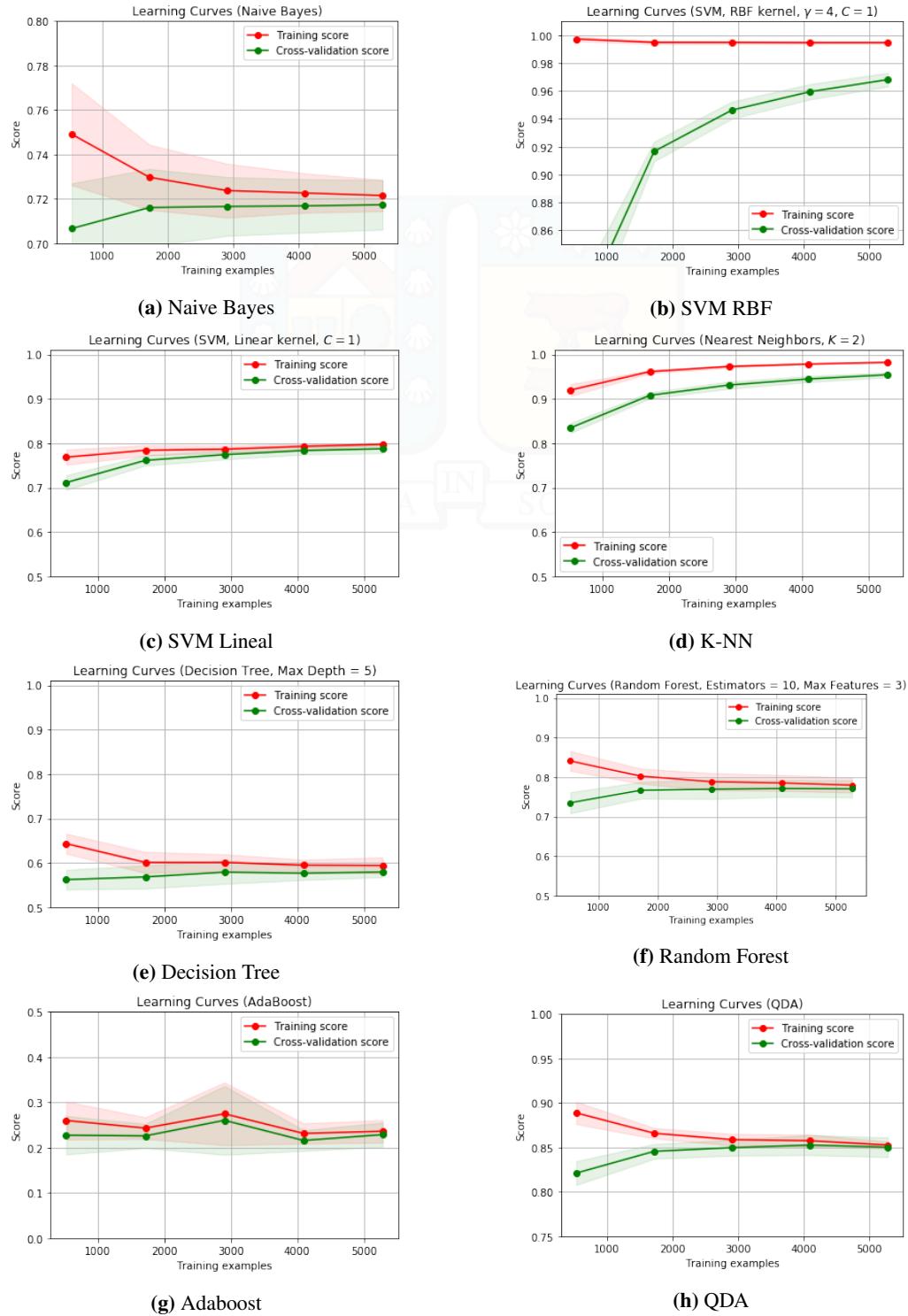


Figura 5.12: Curvas de aprendizajes obtenidas por cada clasificador
(Fuente: Elaboración Propia)

modelo múltiples veces, se descarta esta opción por el alto tiempo demandado y se utiliza una separación aleatoria simple en test set y train set, utilizando igual que en los casos anteriores, 20 % de los datos como test set.

Los demás parámetros utilizados para entrenar la red neuronal son un número máximo de pasos o *max-steps* de 20000, que corresponde a los **epoch**, es decir, las veces que la red se recorre hacia adelante y posteriormente se aplica el algoritmo de backpropagation, mientras aprende los parámetros del modelo. Todo ese proceso entonces es lo que se denomina epoch. Además, como optimizador de los parámetros, se utiliza gradiente descendente, con un *learning rate* igual a $\alpha = 0.3$. También se define un *batch size* igual a 32, esto representa que el algoritmo aprenderá paulatinamente en subconjuntos de datos de tamaño igual a 32. Esto se conoce como mini-batch Gradient Descent y el error se calcula sobre cada batch, al igual que la actualización de parámetros. La principal ventaja es que al suministrar datos paulatinamente, la actualización de parámetros es mucho más frecuentemente, lo que ayuda en la convergencia y así no quedar estancado en mínimos locales. Además ayuda a no mantener simultáneamente todos los datos en memoria, por lo que es mucho más eficiente en términos de memoria y procesamiento.

Con respecto a la estructura de la red, esta se configura como una *input layer* de tamaño del número de atributos, en este caso corresponde a 8, debido a los 8 beacons. Posteriormente, vienen las dos capas ocultas y finalmente la *output layer*, la cual tiene como objetivo principal entregar probabilidades a cada clase, es decir, por cada clase retorna un valor entre 0 a 1, y entre todas las clases suman 1, por lo que según el mayor valor se puede determinar la clase más probable. Para entrenar la red se utiliza la *loss function* denominada **cross entropy**, la que es muy utilizada en redes neuronales porque reduce el problema de aprendizaje presentado por la función de costo cuadrática, también muy utilizada en regresión.

La estructura de la red puede visualizarse en la Figura 5.13.

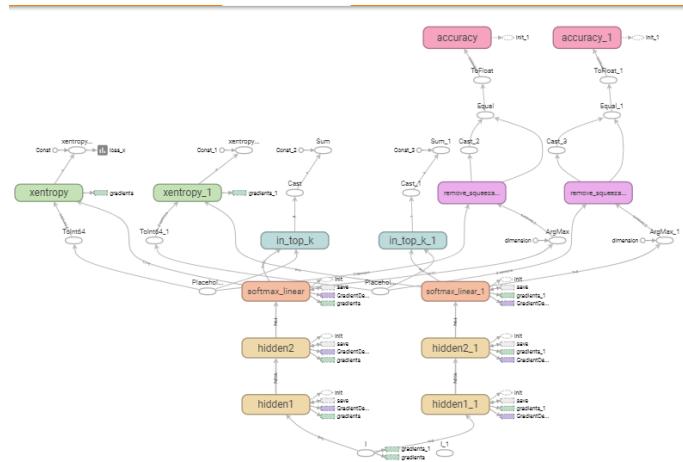


Figura 5.13: Estructura de las redes neuronales construidas. En la figura se muestran ambas, tanto X como Y
(Fuente: Elaboración Propia)

En la imagen se muestra la estructura de la red construida y extraída del visor **Tensorboard**, un addon de Tensorflow para la visualización de gráficos, grafos, histogramas, entre otros. Se debe notar que existen dos redes paralelas, esto se debe a que las redes neuronales para predecir la clase de X y la clase de Y se entranan simultáneamente, por esto aparecen juntas en la imagen, pero cada una tiene su propio entrenamiento y evaluación.

Con todos estos puntos definidos, se muestra los resultados obtenidos en el entrenamiento de la red neuronal profunda. La Figura 5.14 exhibe esta situación.

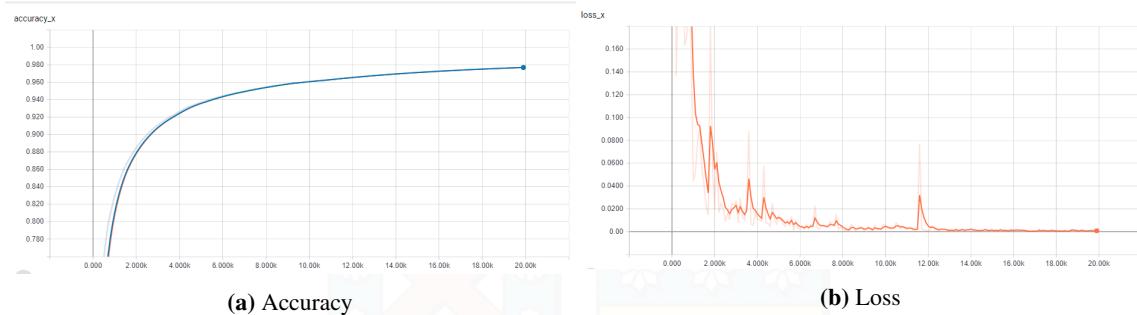


Figura 5.14: Accuracy y Loss(Costo) obtenidos en el entrenamiento de la red neuronal profunda con dos capas ocultas.

(Fuente: Elaboración Propia)

Debe tenerse en cuenta que para la accuracy, los resultados de entrenamiento y pruebas están en la imagen simultáneamente, pero los resultados son tan similares que se sobreponen uno por encima de otro, sin embargo la accuracy sobre el train set es siempre superior(aunque por muy poco) a la accuracy del test set. La función de costo o *Loss function* muestra como la red va aprendiendo a medida de los epoch, ya que lo que se busca es minimizar este valor, el cual expresa el costo del modelo o que tan distante es este a los resultados esperados. Como se menciona anteriormente, esta función de costo corresponde a cross-entropy.

Con toda la información presentada, es claro que los mejores resultados se presentan principalmente en 3 algoritmos, estos son KNN, SVM con kernel RBF y Redes Neuronales profundas. Lo anterior demuestra que los datos efectivamente tienen estructuras no lineales, ya que K-NN no depende específicamente de la estructura de los datos, ya que busca un radio o vecindario de clasificación. SVM RBF define áreas de clasificación que precisamente son no lineales, por lo que puede buscar mejores representaciones en este tipo de estructuras que los clasificadores lineales. La forma en que modela estas fronteras no lineales es debido a la elección del kernel RBF el cual por su función de radio basal es flexible en torno a este tipo de regiones. Por último, las redes neuronales pueden aprender de datos no lineales debido a sus funciones de activación que precisamente son no lineales, además de poder aprender patrones más detallados en cada capa, por su capacidad de reducir la complejidad y dimensionalidad de los datos.

Para corroborar los resultados visualizados en los gráficos, se realiza la Tabla 5.1 con los mejores valores de accuracy para cada algoritmo. Aunque esto puede ser suficiente, además se incorpora el mejor error obtenido en metros, es decir, el error cuadrático medio para cada coordenada respectivamente, ya que, aunque la accuracy determina que tanto acierta el algoritmo, igualmente es necesario establecer el error medio, porque es preferible tener un valor de error más pequeño, debido a que este representa el error presente al momento de la fase online de ubicación. Este error esta expresado en metros y se separa según coordenadas, ya sea X o Y.

Tabla 5.1: Tabla de resultados algoritmos de clasificación

Algoritmo	Accuracy	Error medio X	Error medio Y	Error Absoluto
NN	97.94 %	0.1579	0.0735	0.1741
SVM(RBF, $C = 1, \gamma = 4$)	96.81 %	0.2254	0.1018	0.2473
KNN($k = 2$)	95.43 %	0.9842	0.1575	0.9967
QDA	85.25 %	5.1103	4.7175	6.9548
SVM(Lineal, $C = 1$)	78.75 %	9.3163	6.0387	11.1022
Random Forest	77.33 %	11.3430	3.1409	11.7698
Naive Bayes	71.73 %	12.2303	9.4836	15.4763
Decision Tree(max depth = 5)	57.91 %	57.8012	5.6412	58.0758
Adaboost	26.03 %	150.8848	6.5333	151.0261

Efectivamente los mejores clasificadores en términos de accuracy también lo son en error medio, ya sea absoluto o por coordenadas. Para el error absoluto se utiliza la fórmula de distancia punto a punto, que es lo más natural debido a la naturaleza del problema.

Con lo anterior, se decide utilizar los tres mejores algoritmos, es decir, que presentan buenos resultados ya sea en accuracy o en error de distancia, los cuales son Red neuronal profunda de dos capas escondidas, SVM con kernel RBF y K-NN. A continuación, es necesario realizar el análisis de reducción de la dimensión y quitar la correlación de las predictoras mediante el uso de PCA.

Además del análisis de resultados, es necesario establecer el costo asociado a configurar los hiperparámetros de cada algoritmo, y además el tiempo asociado a su entrenamiento. En primera instancia se debe tener en cuenta que los parámetros de los clasificadores son buscados en grilla previamente, lo cual supone un tiempo mayor de entrenamiento, pero sin embargo esto es automatizado para encontrar los parámetros que obtengan los mejores resultados. Por lo anterior, el costo de entrenamiento de los clasificadores utilizando scikit no supone un problema. Además, existen algoritmos que requieren configuración de muchos menos parámetros, y esto es crucial en el problema de localización, ya que al ser cada recinto distinto, es sumamente importante obtener los parámetros adecuados. Por ejemplo, K-NN solo requiere configuración del parámetro estructural k , por lo que el tiempo de configuración se ve ampliamente reducido con respecto a otros clasificadores como NN que además de requerir la configuración de los hiperparámetros estructurales, también necesita configurar otro tipo de parámetros asociados al algoritmo, como por ejemplo el *learning rate*, numero de epoch, entre otros.

Todos los costos anteriores deben ser considerados, ya que una búsqueda exhaustiva de hiperparámetros requiere tiempo y trabajo humano dedicado a esa labor, por lo que, si el recinto de clasificación crece significativamente, se necesita mucho más esfuerzo en la configuración. Esto puede parecer irrelevante en la clasificación, pero son costos que se deben considerar, en esta suerte de trade-off debido a la perdida de precisión en los clasificadores por malos hiperparámetros, versus el costo de configuración previa de ellos en la fase de entrenamiento. En este caso, todos los clasificadores obtienen los resultados mostrados en unos cuantos minutos, esto se debe a los pocos features del dataset, por lo que el entrenamiento se torna rápido, sin embargo, Neural Network requiere mucho más tiempo de configuración y entrenamiento, esto es alrededor de 3 horas de entrenamiento utilizando los 20000 epoch mencionados, todo esto configurando manualmente los parámetros, ya que en general para deep learning no existen métodos certeros de búsqueda de parámetros, ya que búsqueda en grilla demoraría mucho, por lo que las heurísticas no funcionan muy bien y la configuración debe ser exhaustiva por parte del experimentador.

5.8. Entrenamiento utilizando PCA

Para el uso de PCA, a pesar de que ya se tiene una noción de que clasificadores funcionan mejor en este tipo de problemas, igualmente se analiza para los otros tipos de clasificadores, ya que de esta manera se puede evaluar el efecto de aplicarlo en estos, además sirven como guía para el análisis posterior.

Lo primero a determinar claramente, es el número de componentes principales que deben ser utilizadas para disminuir los tiempos de procesamiento y el número de componentes no ortogonales, es decir, reducir la información redundante total. Para ello se debe seleccionar el mínimo número de componentes principales, ya que mientras más se añaden, es mucho más probable que los datos presenten información no relevante, es decir, ruido e información duplicada. Obviamente, es esencial reducir este tipo de datos para así mejorar el desempeño de los algoritmos de clasificación.

Para la selección, es necesario realizar pruebas que ayuden a la toma de decisiones sobre el número de componentes principales a utilizar, ya que hasta el momento no existe un algoritmo que lo determine automáticamente, por lo que depende netamente del experimentador y los resultados deseados en términos de accuracy y ahorro de procesamiento. Para ello se proponen tres métodos explicados a continuación. El primer método se basa en la información contextual presente en cada componente principal a través de los valores propios de la matriz de covarianzas. Estos valores propios representan que tanto se explica la varianza en una determinada dirección, en donde la dirección es la determinada por el vector propio asociado al valor propio de esa componente principal. Por lo anterior, a medida que los valores propios de una determinada componente principal incrementan su valor, esta componente principal tiende a tener más información valiosa en términos generales para el conjunto total de datos.

Después de aplicar el algoritmo PCA, el espacio de características se mantiene constante según el número de componentes originales, por lo que si el radiomap se proyecta directamente no hay mejoría y la complejidad computacional incluso puede aumentar. Para la selección, se incorpora una variable U , donde esta representa el número de componentes principales seleccionadas. Como se explica anteriormente, el primer método es determinar los valores propios que aportan más información(varianza), mientras más alto su valor, la componente principal asociada mantiene más información valiosa para los datos, es decir, puede interpretarse como la importancia de esa componente. Para determinar los valores propios más relevantes, estos se determinan según un criterio muy conocido, el cual es seleccionar aquellos valores propios con valor cercano o mayor a 1. Para ello, se grafican los valores propios en orden descendente en la Figura 5.15

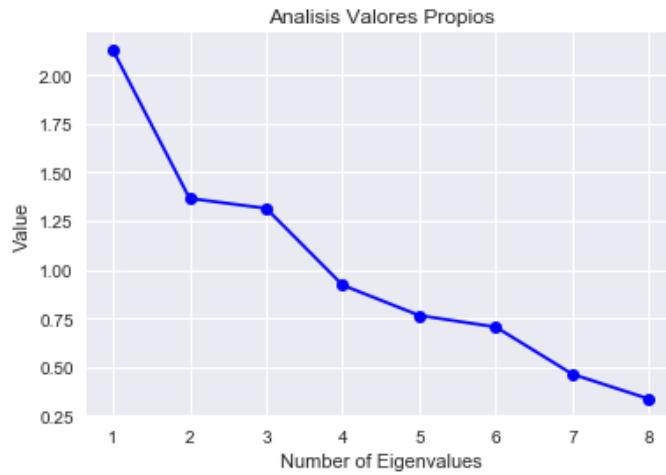


Figura 5.15: Análisis de los valores propios obtenidos a partir de la matriz de covarianza utilizada en el algoritmo PCA
(Fuente: Elaboración Propia)

Como se observa, los primeros 4 valores propios cumplen la heurística de selección necesaria, por lo que mantienen la mayor parte de la información del *dataset*.

Para el segundo método, es necesario establecer la suma acumulada porcentual de la varianza explicada, lo cual es sinónimo de el porcentaje de información relevante retenida o acumulada. Para ello, es necesario igualmente utilizar los valores propios que representan la varianza para la selección. La fórmula a utilizar entonces es:

$$\frac{\sum_{i=1}^U \lambda_i}{\sum_{i=1}^M \lambda_i} > \xi \quad (5.3)$$

De donde $\{\lambda_1, \lambda_2, \dots, \lambda_M\}$ corresponden a los valores propios y :

$$\frac{\sum_{i=1}^U \lambda_i}{\sum_{i=1}^M \lambda_i} \quad (5.4)$$

Representa la suma acumulada que retiene la información. Además, el límite requerido es ξ . Este límite representa la cantidad de información mínima con las cual se acepta el criterio, y habitualmente se elige un valor entre 70 a 80 %. Para esto se muestra la Figura 5.16 la cual describe esta situación.

Como se aprecia en la Figura 5.16, las primeras 5 componentes principales retienen la cantidad de información especificada según el límite impuesto $\xi = 0.8$. Para clarificar este punto se construye una tabla resumen con los valores propios, la proporción de varianza explicada y la suma acumulada de estas proporciones.

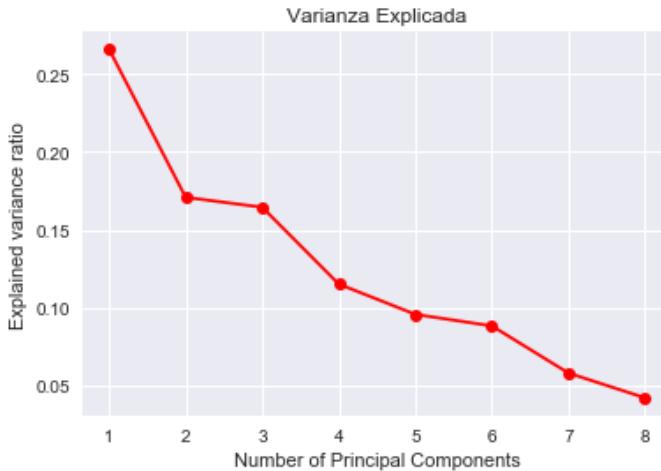


Figura 5.16: Proporción de la varianza explicada respecto a la varianza total para cada componente ordenada por su valor en orden descendente

(Fuente: Elaboración Propia)

Tabla 5.2: Tabla resumen varianza explicada y acumulada PCA

Indicador	PC1	PC2	PC3	PC4	PC5	PC6	PC7	PC8
Valor Propio	2.1271	1.3665	1.3154	0.92088	0.7652	0.7058	0.4637	0.3363
Proporción Varianza Explicada	0.2658	0.1707	0.1644	0.1150	0.0956	0.0882	0.0579	0.0420
Suma Acumulada Proporción	0.2658	0.4366	0.6010	0.7161	0.8117	0.9000	0.9579	1.000

Con respecto al tercer método, este se basa en seleccionar las primeras componentes principales según los resultados obtenidos en los clasificadores, por lo que se necesita PCA ya aplicado, no solo basta con la matriz de correlación, si no que los algoritmos completos. Para realizar este método entonces se decide aplicar PCA en todos los algoritmos de clasificación nombrados anteriormente, con ellos se pretende establecer los mejores valores de accuracy y en qué puntos a nivel de componentes principales, la mayoría de los clasificadores establecen sus mejores puntuaciones en términos de accuracy, por lo que añadir más componentes no provoca una mejoría significativa.

La Figura 5.17 muestra la comparativa de los clasificadores.

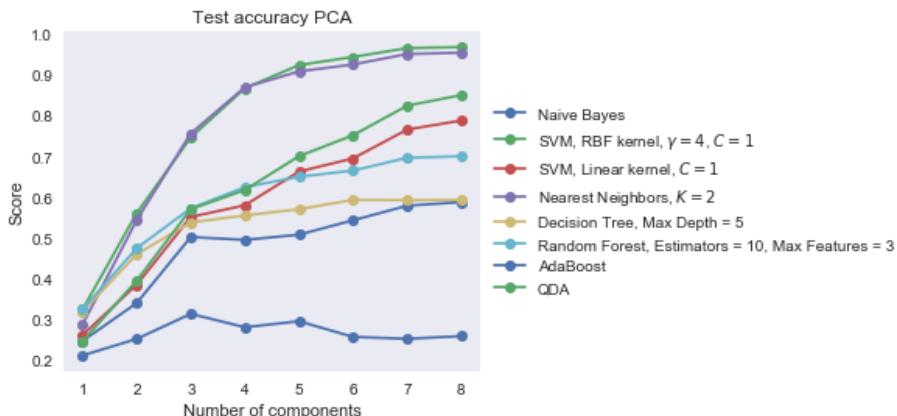


Figura 5.17: Comparativa de resultados obtenidos por los clasificadores utilizando PCA

(Fuente: Elaboración Propia)

Según la comparativa, es claro que la mayoría de los clasificadores presentan una gran mejoría hasta la quinta componente, y posteriormente la ganancia no es significante respecto a la información aportada a los respectivos algoritmos. Esto indica al igual que los otros métodos de selección, que a partir de la quinta componente la varianza aportada no es realmente significativa, por lo que puede reducirse la dimensionalidad de los datos a 5, ya que los tres métodos utilizados presentan resultados similares y coinciden en este valor de componentes principales.

Posteriormente, se procede a evaluar según el procedimiento realizado anteriormente para los clasificadores sin utilizar PCA. Los resultados obtenidos en cada clasificador se muestran en la Figura 5.18. Se debe destacar que los hiperparámetros de cada algoritmo no fueron cambiados respecto a sus versiones sin utilizar PCA.



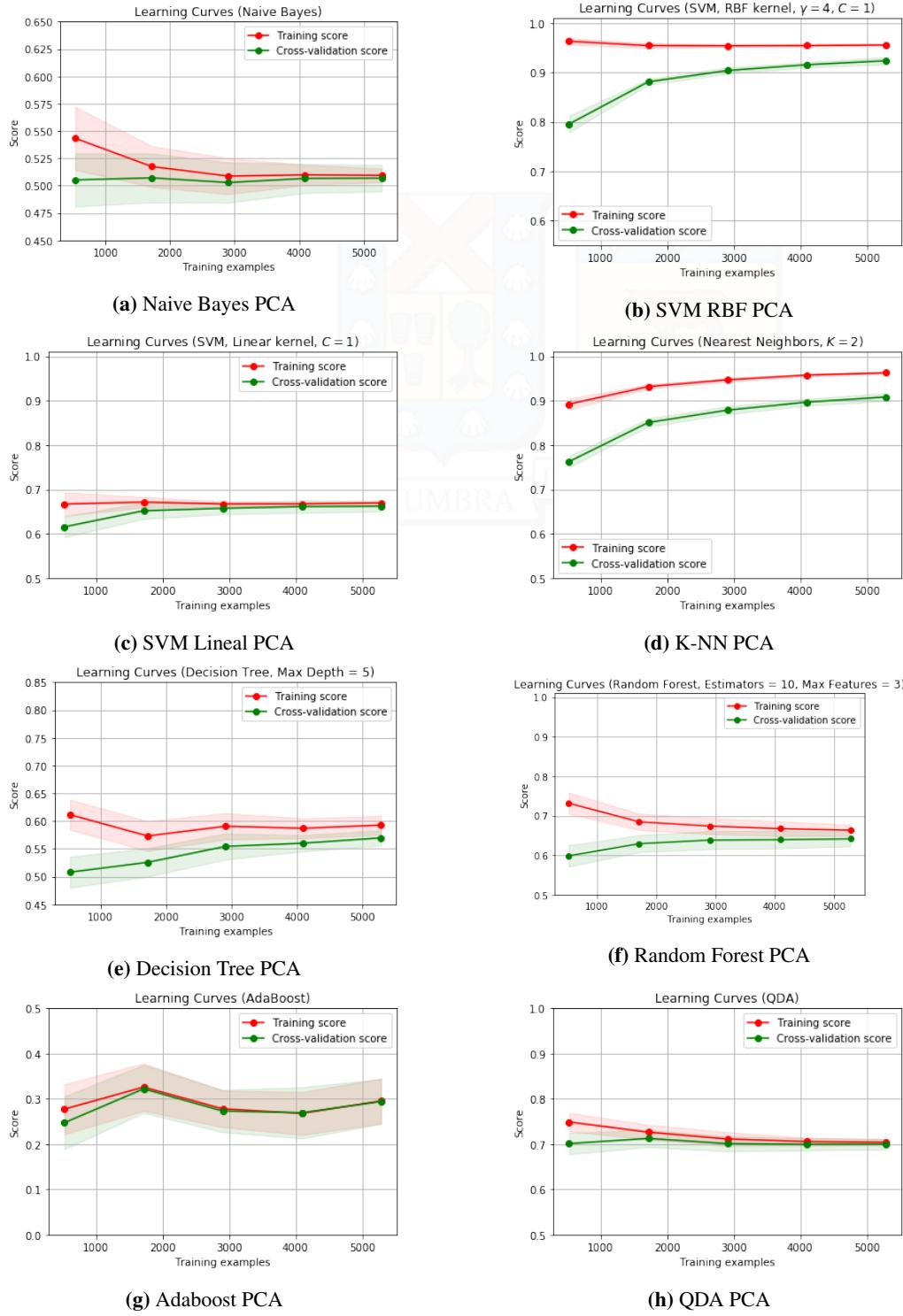


Figura 5.18: Curvas de aprendizajes obtenidas por cada clasificador utilizando PCA con 5 componentes
(Fuente: Elaboración Propia)

En este caso, se obtienen resultados similares a los obtenidos sin utilizar PCA, esto es un indicativo de que la elección de las componentes principales es correcta. Para analizar las redes neuronales utilizando PCA se procede a graficar la accuracy y la función de perdida al igual que en casos anteriores, como indica la Figura 5.19 .

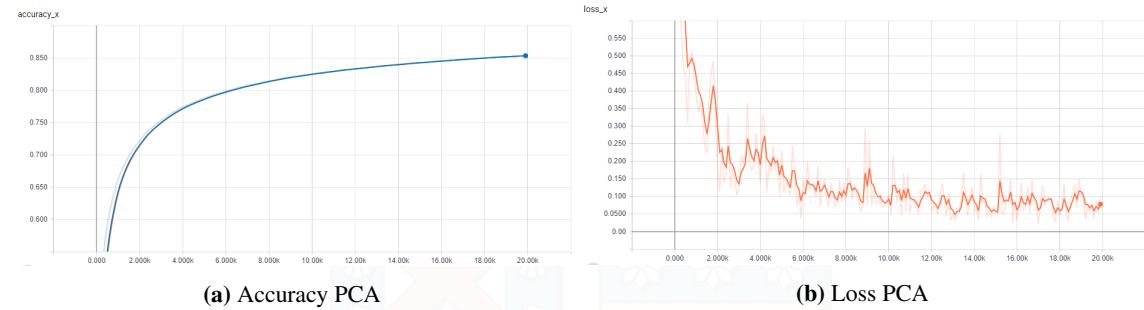


Figura 5.19: Accuracy y Loss(Costo) obtenidos en el entrenamiento de la red neuronal profunda con dos capas ocultas utilizando PCA con 5 componentes principales.

(Fuente: Elaboración Propia)

Al igual que los clasificadores convencionales, los resultados son muy similares a los obtenidos mediante clasificadores sin utilizar PCA. Esto es un indicativo de que la elección de las componentes principales es acertada. Para corroborar esto se construye la tabla Tabla 5.3 la cual es símil a la Tabla 5.1

Tabla 5.3: Tabla de resultados algoritmos de clasificación PCA

Algoritmo	Accuracy	Error medio X	Error medio Y	Error Absoluto
NN	93 %	1.8956	0.6589	2.0068
<i>SVM(RBF, C = 1, γ = 4)</i>	92.39 %	2.4581	0.7830	2.5797
<i>KNN(k = 2)</i>	90.83 %	2.1381	0.4872	2.1929
QDA	71.25 %	15.9418	9.3042	18.4583
<i>SVM(Lineal, C = 1)</i>	66.20 %	19.5878	10.4824	22.2162
Random Forest	64.15 %	21.3284	5.4472	22.0130
Decision Tree(max depth = 5)	56.96 %	33.5151	8.9163	34.6808
Naive Bayes	50.71 %	31.3406	10.0727	32.9194
Adaboost	32.20 %	73.9345	9.3042	74.5176

Los valores obtenidos entonces demuestran que a pesar de que la accuracy se reduce solo un poco, los valores de error medio decrementan significativamente en el contexto del problema abordado, ya que para el posicionamiento en interiores se espera un error lo más pequeño posible, ya que esto significa una mejor localización. Por lo anterior, se debe tener en cuenta estos factores al momento de seleccionar las componentes principales, ya que, al perder información, se pierde exactitud en las mediciones, lo que repercute significativamente al momento de realizar pruebas reales.

El comportamiento en términos generales sigue el mismo patrón que al no aplicar técnicas de reducción de dimensionalidad, es decir, los clasificadores mantienen el orden relativo en cuanto a accuracy y nuevamente los clasificadores capaces de distinguir patrones no lineales en los datos son los dominantes, lo cual prueba estas características de los datos. También se debe notar que los tres primeros lugares corresponden nuevamente a redes neuronales profundas, máquina de vectores de soporte con kernel RBF y finalmente vecinos más cercanos con $k = 2$.

Por lo anterior, se decide utilizar los tres clasificadores NN, SVM y KNN para realizar la experimentación en el estacionamiento descrito anteriormente. A continuación, se define el método utilizado para realizar las pruebas.

5.9. Fase Online

Para realizar las pruebas dentro del recinto es necesario establecer el marco a utilizar para determinar la exactitud y precisión del sistema. En primer lugar, se debe tener en cuenta que no hay forma de determinar

el error absoluto, producto de que para ello se debe proporcionar la posición real, lo cual es contradictorio, ya que esto es precisamente lo que se busca. Existen métodos de buscar este error, por ejemplo (Anand, 2014) describe una forma de medir los errores mediante el uso de la creación de caminos predefinidos. Luego para medir la precisión, se infiere la posición estimada cada 10 metros siguiendo la trayectoria definida. Luego, con la posición estimada y la posición teórica determinada por el camino establecido, se puede obtener la distancia que representa el error, la cual está definida mediante el vector perpendicular desde la trayectoria definida hasta el punto predicho.

Otro tipo de enfoque es utilizar el tiempo recorrido y mediante las ecuaciones de movimiento obtener una posición aproximada que puede ser comparada con la inferida por los algoritmos de máquinas de aprendizaje.

El problema con estos enfoques es que se basan en propiedades del experimentador, como por ejemplo utilizar sensores basados en detección de pasos, lo cual no es del todo preciso, o también al utilizar las ecuaciones de posición se requiere que la velocidad del experimentador sea constante y medir exactamente en un instante indicado, lo cual dificulta mucho las pruebas y también agrega factores de error externos lo que se traduce en resultados ruidosos y poco fidedignos. Entonces lo que se propone para determinar los resultados son dos formas llamadas método estático y método dinámico. En el método estático lo que se hace es permanecer quieto en un determinado punto durante un tiempo predefinido. El tiempo utilizado en este caso corresponde a 15 minutos, y el punto es específico es (22, 12). Para el caso del método dinámico, lo que se busca es abarcar la mayor cantidad de puntos posibles y establecer de ante mano el punto en el cual se realizaran las mediciones a través de una caminata. En este caso se decide hacer una caminata a través de todos los 44 puntos posibles de izquierda a derecha, arriba a abajo durante un tiempo de 30 minutos.

Para realizar las pruebas se adiciona una pantalla a la aplicación precisamente para la fase Online, la Figura 5.20 muestra esta situación.



Figura 5.20: Pantallas utilizadas en la fase Online para la realización de las pruebas de experimentos reales
(Fuente: Elaboración Propia)

Como se observa en la imagen Figura 5.20, en primer lugar se debe elegir el algoritmo a utilizar en el posicionamiento, luego seleccionar el patrón para elegir el punto en donde se realizaran las pruebas y finalmente iniciar, lo cual comienza a posicionar al usuario en tiempo real. Para el método dinámico los pasos son igual, solo se debe utilizar el *checkbox* al lado del botón iniciar y cambiar progresivamente el punto a utilizar mientras se sigue el camino de predicción.

A pesar de que se selecciona un algoritmo para la visualización del posicionamiento, internamente la aplicación ejecuta todos los algoritmos mostrados, es decir NN, SVM y KNN con sus respectivas versiones utilizando PCA. El fin de esto es ahorrar tiempo, ya que de esta forma se pueden obtener los resultados

para todos los algoritmos concurrentemente sin necesidad de ejecutarlos de manera secuencial. Luego la aplicación tiene la capacidad de crear archivos de *log*, es decir, guardar los resultados obtenidos. El formato de estos archivos es diseñado para su posterior análisis. La primera línea contiene el tipo de método, estático o dinámico. La siguiente línea de texto muestra el punto teórico con el cual se comparan los datos. Posteriormente el nombre del algoritmo, luego una lista de valores de la posición *X*, a continuación, una lista de valores *Y* y finalmente el tiempo de ejecución del algoritmo. Para cada algoritmo se repite esta estructura, en el orden mostrado en la Figura 5.20a. Claramente la lista de valores *X* e *Y* tienen las mismas dimensiones y forman las predicciones finales según su posición relativa, es decir, el primer elemento de la lista *X*₁ con el primer elemento de la lista *Y*₁ forman el primer punto predicho por el algoritmo (*X*₁, *Y*₁).

Para clarificar esto, la Figura 5.21 muestra un extracto de un archivo obtenido luego de realizar la experimentación de la fase online. Este archivo es de tipo dinámico, con lo que contiene múltiples puntos de clasificación.

```
Online
2.0 8.0
KNN Original
18.0 10.0 10.0 10.0 10.0 10.0 10.0 10.0 10.0 10.0 10.0 10.0 10.0 10.0 10.0 10.0 10.0 10.0
10.0 10.0 10.0 10.0 10.0 10.0 10.0 10.0 10.0 10.0 10.0 10.0 10.0 10.0 10.0 10.0 10.0 10.0
10.0 10.0 10.0 10.0 10.0 10.0 10.0 10.0 10.0 10.0 10.0 10.0 10.0 10.0 10.0 10.0 10.0 10.0
10.0 10.0 10.0 10.0 10.0 10.0 10.0 10.0 10.0 10.0 10.0 10.0 10.0 10.0 10.0 10.0 10.0 10.0
10.0 10.0 10.0 10.0 10.0 10.0 10.0 10.0 10.0 10.0 10.0 10.0 10.0 10.0 10.0 10.0 10.0 10.0
10.0 10.0 10.0 10.0 10.0 10.0 10.0 10.0 10.0 10.0 10.0 10.0 10.0 10.0 10.0 10.0 10.0 10.0
10.0 10.0 10.0 10.0 10.0 10.0 10.0 10.0 10.0 10.0 10.0 10.0 10.0 10.0 10.0 10.0 10.0 10.0
10.0 10.0 2.0 2.0 2.0 2.0 2.0 2.0 2.0 2.0 2.0 2.0 2.0 2.0 2.0 2.0 2.0 10.0
10.0 10.0 10.0 10.0 10.0 10.0 10.0 10.0 10.0 10.0 10.0 10.0 10.0 10.0 10.0 10.0 10.0 10.0
10.0 10.0 10.0 10.0 10.0 10.0 10.0 10.0 10.0 10.0 10.0 10.0 10.0 10.0 10.0 10.0 10.0 10.0
10.0 10.0 10.0 10.0 10.0 10.0 10.0 10.0 10.0 10.0 10.0 10.0 10.0 10.0 10.0 10.0 10.0 10.0
10.0 10.0 10.0 10.0 10.0 10.0 10.0 10.0 10.0 10.0 10.0 10.0 10.0 10.0 10.0 10.0 10.0 10.0
10.0 10.0 20.0 20.0 20.0 20.0 20.0 20.0 20.0 20.0 20.0 20.0 20.0 20.0 20.0 20.0 20.0 20.0
20.0 20.0 20.0 20.0 20.0 20.0 20.0 20.0 20.0 20.0 20.0 20.0 20.0 20.0 20.0 20.0 20.0 20.0
20.0 20.0 20.0 20.0 20.0 20.0 20.0 20.0 20.0 20.0 20.0 20.0 20.0 20.0 20.0 20.0 20.0 20.0
20.0 20.0 20.0 20.0 20.0 20.0 20.0 20.0 20.0 20.0 20.0 20.0 20.0 20.0 20.0 20.0 20.0 20.0
20.0 20.0 20.0 20.0 20.0 20.0 20.0 20.0 20.0 20.0 20.0 20.0 20.0 20.0 20.0 20.0 20.0 20.0
20.0 20.0 20.0 20.0 20.0 20.0 20.0 20.0 20.0 20.0 20.0 20.0 20.0 20.0 20.0 20.0 20.0 20.0
20.0 20.0 20.0 20.0 20.0 20.0 20.0 20.0 20.0 20.0 20.0 20.0 20.0 20.0 20.0 20.0 20.0 20.0
20.0 20.0 20.0 20.0 20.0 20.0 20.0 20.0 20.0 20.0 20.0 20.0 20.0 20.0 20.0 20.0 20.0 20.0
65.81553398058253
KNN PCA
18.0 10.0 10.0 10.0 10.0 10.0 10.0 10.0 10.0 10.0 10.0 10.0 10.0 10.0 10.0 10.0 10.0 10.0
10.0 10.0 10.0 10.0 10.0 10.0 10.0 10.0 10.0 10.0 10.0 10.0 10.0 10.0 10.0 10.0 10.0 10.0
```

Figura 5.21: Archivo obtenido en la experimentación online

(Fuente: Elaboración Propia)



6 | Resultados

6.1. Métricas Obtenidas

Para los resultados, se deben elaborar métricas que permitan determinar cuáles fueron los mejores algoritmos, es decir, los que presentan mejor desempeño según la experimentación determinada anteriormente. Primero se construyen las tablas que muestran los valores más relevantes en primera instancia, es decir, los errores en cada eje coordenado, sus promedios de error, varianzas y finalmente el *root squared mean error*, o error cuadrático medio, el cual en términos euclidianos, representa la distancia de un punto a otro. La fórmula de este último es:

$$RMSE = \sqrt{\frac{1}{n} \sum_{j=1}^n (y_j - \hat{y}_j)^2} \quad (6.1)$$

Para las tablas se considera el método dinámico y estático definido anteriormente. La Tabla 6.1 muestra los resultados obtenidos en el método dinámico sin utilizar PCA. Se debe considerar que todos los valores están en metros.

Tabla 6.1: Resultados método dinámico sin utilizar PCA

Clasificador	Error x	Error y	Varianza x	Varianza y	RMSE
KNN	1.5858	4.6391	7.1970	2.1780	6.9323
SVM	6.8207	2.8874	1.7243	0.5989	10.0323
NN	4.3784	3.9113	13.0950	5.0712	8.2994

Como muestra la Tabla 6.1, los mejores resultados en términos generales son obtenidos por el método KNN, ya que los promedios en general son bajos, así como el RMSE. A pesar de esto, se debe notar que lo sigue Neural Network con un RMSE promedio de 8.2994 metros.

La Tabla 6.2 muestra los resultados obtenidos por el método dinámico utilizando PCA.

Tabla 6.2: Resultados método dinámico utilizando PCA con 5 componentes

Clasificador	Error x	Error y	Varianza x	Varianza y	RMSE
KNN PCA	2.0023	4.3983	7.5113	2.0696	6.6812
SVM PCA	6.8948	2.4257	4.1134	2.0348	9.5668
NN PCA	5.8874	4.4513	8.6088	3.2089	9.5188

En este caso, los resultados son ligeramente mejores en términos de RMSE, lo cual indica que PCA si puede eliminar parte de la correlación espacial e información repetida dentro de los datos. En este caso nuevamente KNN con PCA obtiene el mejor resultado, seguido por Neural Network y finalmente SVM. Las varianzas de cada eje igualmente son relativamente altas, por lo que esto puede afectar mucho

el posicionamiento en general, ya que, a mayor varianza, más error durante el tiempo se presenta, y más alejados están los valores desde el promedio.

Para la siguiente tabla, ahora se considera el método estático, el cual como se ha descrito, mide la posición al estar estático en un determinado lugar.

Tabla 6.3: Resultados método estático sin utilizar PCA

Clasificador	Error x	Error y	Varianza x	Varianza y	RMSE
KNN	3.2385	1.5417	3.3321	1.0940	5.0520
SVM	5.2493	1.6986	2.0598	0.7594	7.0241
NN	3.7300	2.1937	5.8885	0.7373	4.4857

Para este caso, la Tabla 6.3, muestra que ahora los mejores resultados son obtenidos por el método Neural Networks con un RMSE de 4.4857 en promedio. Para todos los algoritmos, las varianzas se ven reducidas sustancialmente, lo que es esperable debido a que, al estar estático en una posición, las ondas electromagnéticas no se ven alteradas o cambiantes a través del tiempo, lo cual estabiliza la señal recibida por los Beacons. Para este caso, primero es NN, luego KNN y finalmente SVM en términos de RMSE. Igualmente, los valores de error obtenidos son muy altos para lograr un posicionamiento exacto.

Por último, las métricas obtenidas para el posicionamiento estático utilizando PCA se muestran a continuación.

Tabla 6.4: Resultados método estático utilizando PCA con 5 componentes

Clasificador	Error x	Error y	Varianza x	Varianza y	RMSE
KNN PCA	2.9892	1.4475	3.1487	1.6391	5.0340
SVM PCA	3.2313	1.4905	3.0630	1.7817	5.1757
NN PCA	1.5578	1.7488	3.8045	2.6885	3.9341

La Tabla 6.4 muestra mejores resultados para todos los clasificadores, disminuyendo así los valores totales. Esto muestra nuevamente que mediante la técnica PCA es posible reducir el error. Por otra parte, las varianzas permanecen pequeñas a excepción de Neural Network. El mejor resultado es obtenido por Neural Network con PCA con un error promedio de 3.9341 metros, seguido por KNN PCA y finalmente SVM PCA.

El siguiente análisis realizado, tiene que ver con el valor de *Cumulative distribution function*, el cual está definido como el valor de una variable aleatoria X , o su función de distribución, que al ser evaluada en x , es la probabilidad que X tome valores menores o iguales a x . Esta función está definida matemáticamente como:

$$F_X(x) = P(X \leq x) \quad (6.2)$$

La importancia de este indicador es que refleja en este caso, que tan frecuente se repite un valor de error, o, dicho de otra forma, en qué porcentaje se puede asegurar que el error será menor o igual a un determinado valor, por ejemplo el 90 % de las veces el error es menor a 10 metros. Con esta definición, a continuación, se procede a mostrar los gráficos obtenidos tanto para el método estático como para el método dinámico. La Figura 6.1 muestra esta situación.

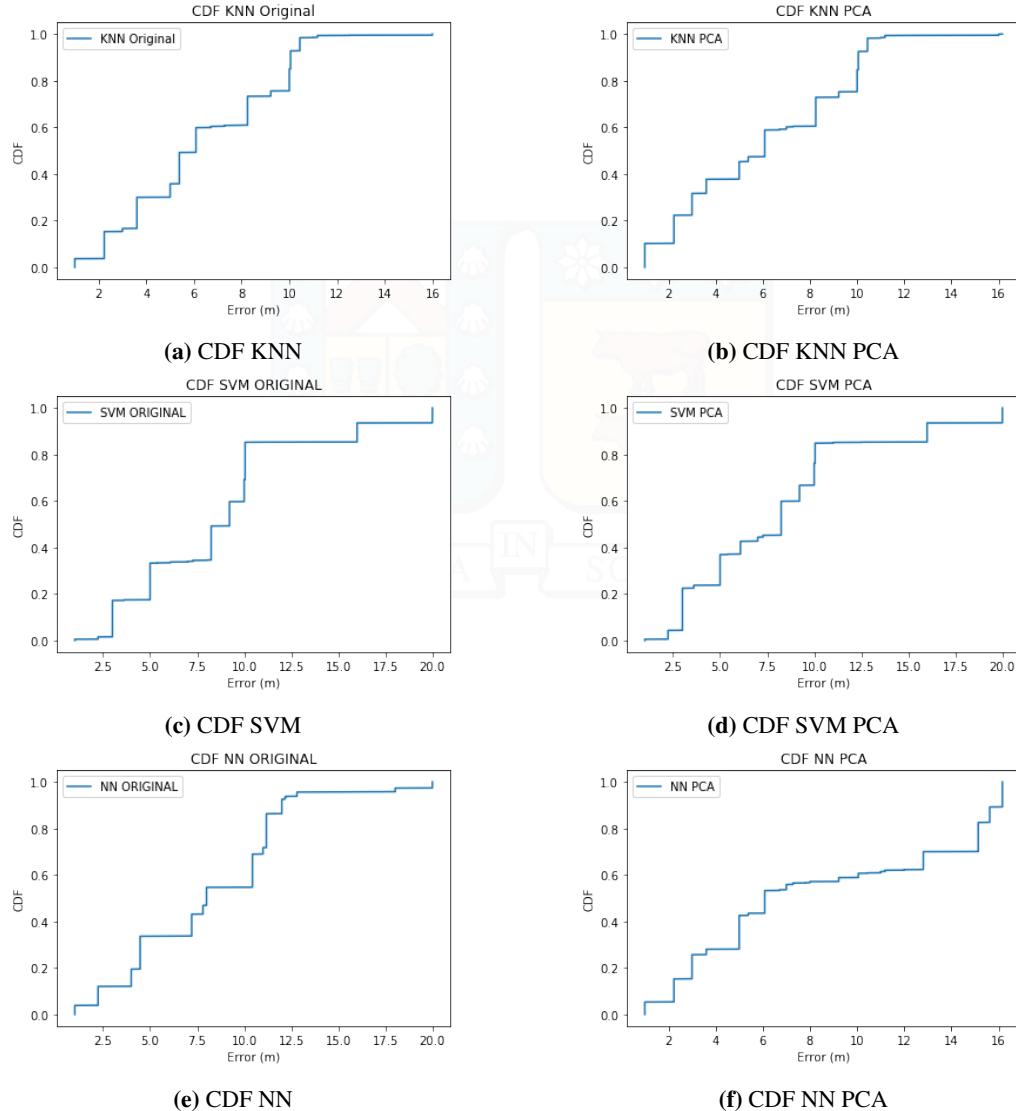


Figura 6.1: Cummulative distribution function para el método dinámico obtenido utilizando los distintos algoritmos
(Fuente: Elaboración Propia)

Los gráficos muestran que los valores son muy similares en cuanto a si se utiliza PCA o no. K-NN se mantiene el 80 % de las veces bajo los 10 metros, en ambos casos y cerca del 60 % bajo los 6 metros. Por otro lado, SVM igualmente el 80 % de las veces está bajo los 10 metros con respecto a SVM sin utilizar PCA, el 50 % de las veces está bajo los 8 metros y utilizando PCA esto ocurre el 60 % de las veces por lo que PCA es más estable en este sentido, y genera valores más uniformes. Finalmente, con las redes neuronales, los resultados presentados sin utilizar PCA, el 80 % de las veces esta cercano a los 12 metros y 50 % de las veces en los 7.5 metros. Utilizando PCA en este último caso, los resultados son mejores para porcentajes más bajos, ya que el 80 % de las veces esta sobre los 15 metros aproximadamente, pero esto no cambia relativamente hacia porcentajes superiores, sin embargo para porcentajes inferiores los valores mejoran, ya que por ejemplo para un error menor a 6 metros, ocurre con una probabilidad del 60 % .

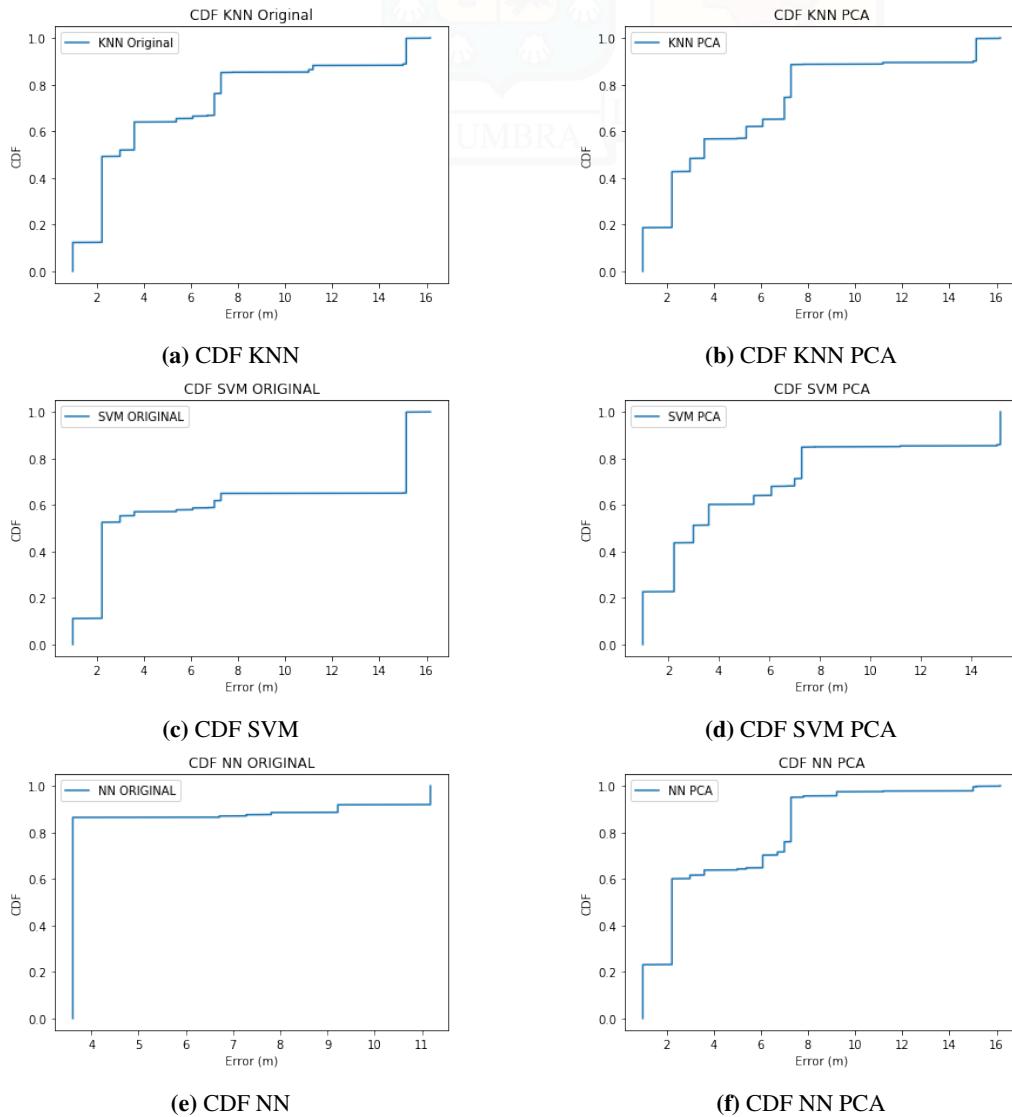
Para ver el mínimo error al 100 %, es decir, el error mínimo que está asegurado, se realiza la Tabla 6.5 comparando sus valores utilizando o no PCA. Nuevamente los valores están en metros.

Tabla 6.5: Error con un CDF del 100 % para los algoritmos analizados en el método dinámico

Clasificador	Sin PCA	Con PCA	Mejora
KNN	16.576	16.1554	2.5374 %
SVM	20.8962	19.3874	7.2204 %
NN	20.5677	16.1554	21.4525 %

Como se observa en la Tabla 6.5, todos los valores son mejorados utilizando PCA, además los errores menores se obtienen con NN y KNN, aunque KNN es más estable ya que con y sin PCA presenta valores semejantes.

Para el método estático se realiza el mismo análisis. Los gráficos obtenidos para el error CDF son los siguientes:

**Figura 6.2:** Cummulative distribution function para el método estático obtenido utilizando los distintos algoritmos
(Fuente: Elaboración Propia)

Los gráficos para los casos estáticos muestran de manera general que los resultados son mucho mejores, esto es esperable debido a como se discute anteriormente, al estar estático es mucho más estable la

señal y por ello se obtienen mejores valores de CDF. Para KNN, el error a los 8 metros se mantiene con una probabilidad de sobre el 80 %, lo cual representa una mejora aproximada de 2 metros respecto al método dinámico. Utilizando PCA se mantiene la tendencia, además se debe observar que para un error menor o igual a 4 metros, ocurre con probabilidad de 60 % lo cual es una mejora significativa.

Para SVM, aunque el error es muy grande en general, se debe tener en cuenta que con cerca del 50 % de las veces, el error está cercano a los 2 metros. Para SVM con PCA los resultados son muy similares al método dinámico. Finalmente, para las redes neuronales, ocurre un comportamiento distinto a los anteriormente vistos, ya que la probabilidad de que el error sea menor o igual a 2 metros sobrepasa el 80 %, es decir esta siempre muy cercano a la posición verdadera. Esto indica que NN estático es muy estable, a diferencia de lo que ocurre con este mismo algoritmo en el método dinámico. Con respecto a NN con PCA, es claro que no existe mejora con respecto al caso sin PCA, de hecho, empeora el desempeño. Esto puede deberse a factores que serán explicados posteriormente, con respecto a las propiedades de las redes neuronales. Lo que se debe tener en cuenta es que la probabilidad de un error menor a 8 metros es cercana al 90 %, y el 60 % de las veces el error es menor a 2 metros, para NN con PCA.

La tabla del error mínimo al 100 % de CDF se muestra a continuación:

Tabla 6.6: Error con un CDF del 100 % para los algoritmos analizados en el método estático

Clasificador	Sin PCA	Con PCA	Cambio
KNN	16.1554	16.1554	0 %
SVM	16.1554	15.1327	6.33 %
NN	11.18033	16.1554	-44.49 %

Los cambios para KNN y SVM no son significativos, y además los errores mínimos son muy similares, del orden de los 16 metros, lo cual indica que PCA funciona efectivamente para reducir la información redundante. Con respecto a NN, se presenta una anomalía al utilizar o no PCA, esto se debe a que las redes neuronales ya hacen una reducción de la dimensionalidad en sus capas escondidas, y retienen lo mejor de la información en cada paso, por lo que aplicar PCA no ayuda demasiado, debido a que resta información relevante, entonces, para Neural Networks, específicamente deep learning, PCA no es efectivo, y es mejor pasar la mayor cantidad de información **relevante** que se posea.

El siguiente análisis tiene relación con la dispersión del error, vale decir, que tan distintos son los valores, y además determinar en qué valores se agrupan en su mayoría los datos. Para ello se utiliza un análisis de diagrama de cajas o *boxplot*. Con este tipo de análisis se puede visualizar la dispersión y distribución de los datos en términos de cuartiles. En primer lugar, se construyen las siguientes tablas que muestran los valores relevantes para el boxplot. La Tabla 6.7 resume los valores de error obtenidos para el método dinámico.

Tabla 6.7: Tabla resumen de los valores necesarios obtenidos para construir el Boxplot del método dinámico

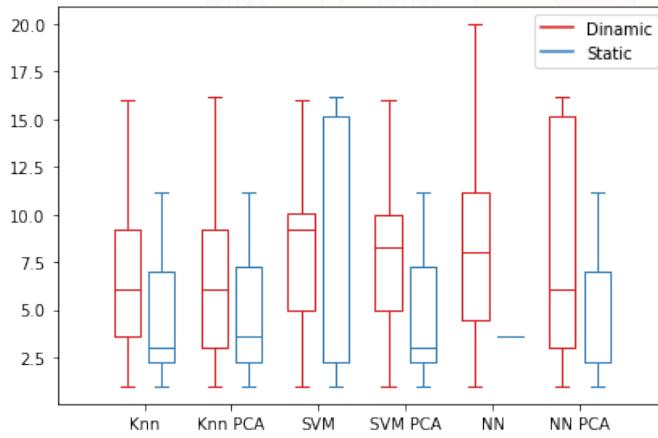
Clasificador	Mediana	25 %	75 %	90 %	95 %	IQR	Min	Max
KNN	6.0827	3.6055	9.2195	10.0498	10.4403	5.6139	0.0	16
KNN PCA	6.0827	3.0	9.2195	10.0498	10.4403	6.2195	0.0	16.1554
SVM	9.2195	5.0	10.0498	16.0	20.0	5.0498	0.0	16.0
SVM PCA	8.2462	5.0	10.0	16.0	20.0	5.0	0.0	16.0
NN	8.0	4.4721	11.1803	12.0	12.8062	6.7082	0.0	20.0
NN PCA	6.0827	3.0	15.1327	16.1554	16.1554	12.1327	0.0	16.1554

Para el método estático, la Tabla 6.8 muestra los valores resultantes de error.

Tabla 6.8: Tabla resumen de los valores necesarios obtenidos para construir el Boxplot del método estático

Clasificador	Mediana	25 %	75 %	90 %	95 %	IQR	Min	Max
KNN	3.0	2.2360	7.0	15.1327	15.1327	4.7639	0.0	11.1803
KNN PCA	3.6055	2.2360	7.2801	15.0	15.1327	5.0440	0.0	11.1803
SVM	2.2360	2.1604	15.1327	15.1327	15.1327	12.9723	0.0	16.1554
SVM PCA	3.0	2.1604	7.2801	15.1327	15.1327	5.1197	0.0	11.1803
NN	3.6055	3.6055	3.6055	9.2195	11.1803	0.0	3.6055	3.6055
NN PCA	2.2360	2.4053	7.0	7.2801	7.2801	4.7639	0.0	11.1803

Finalmente, el diagrama de cajas es el siguiente:

**Figura 6.3:** Diagrama de cajas para método estático y dinámico

(Fuente: Elaboración Propia)

La 6.7 y 6.8 muestran los valores resultantes. Se debe tener en cuenta que los valores son iguales o muy similares cuando se utiliza PCA con el algoritmo original, debido a que la distribución de los datos es idéntica y el algoritmo es el mismo, además como se menciona anteriormente, los valores no cambian significativamente lo cual hace que PCA mantenga la distribución original y no cambie demasiado sus valores.

Se debe notar que muchas veces se repiten los valores de mediana o los cuartiles, por ejemplo, entre distintos algoritmos, sobre todo para el método estático. Esto puede deberse a la gran cantidad de datos.

Por último, antes de analizar en términos generales los resultados, se procede a mostrar las diferencias temporales existentes, es decir, los incrementos referentes a la técnica PCA. La Tabla 6.9 muestra estos resultados en términos de milisegundos y la respectiva mejora.

Tabla 6.9: Mejoras de tiempo en cada algoritmo al usar la técnica PCA para reducir la dimensionalidad

Clasificador	Sin PCA	Con PCA	Incremento
KNN	64.9642	59.6786	8.1361 %
SVM	54.5985	25.6085	53.0966 %
NN	0.7610	0.5777	24.0867 %

6.2. Análisis de resultados

A partir de los resultados obtenidos, se hace el análisis de los mejores algoritmos para el posicionamiento indoor según los parámetros y proceso de experimentación propuesto. Lo primero que se debe tener en consideración son los errores medios, ya que este parámetro afecta directamente al posicionamiento y es uno de los más relevantes al momento de seleccionar el mejor algoritmo. Con respecto al método dinámico, la Tabla 6.1 muestra que el mejor valor es obtenido por KNN con un RMSE de 6.9323 metros, seguido de NN con 8.2994 metros y finalmente SVM con 10.0323. Estos valores reflejan que en este caso KNN sería la mejor opción, sin embargo el error es demasiado grande como para lograr un posicionamiento exacto en los tres casos. Con respecto a la tabla 6.2, esta muestra resultados similares, en este caso KNN nuevamente se impone ante los demás clasificadores, con un error promedio de 6.6812 metros, seguido de redes neuronales con 9.5188 metros y SVM con 9.5668 metros. Se debe tener en cuenta que las varianzas de cada eje coordenado son altas, sobre todo en el eje x , por lo que esto repercute negativamente al momento del posicionamiento y de obtener el error medio **RMSE**. Tomando en consideración los resultados del método dinámico con y sin PCA, claramente KNN es la opción para este tipo de método, en donde existen cambios espaciales, por lo que este no se ve afectado en demasía por la posición en donde es tomada la medición. Esto se debe principalmente a que para KNN es irrelevante los datos de entrada, ya que siempre realiza una búsqueda completa y debe analizar cada punto, por lo que evidentemente puede obtener mejores valores, pero su procesamiento demora más. Además, es mejor utilizar PCA, sobre todo para KNN, así se reduce la información irrelevante y el algoritmo realiza menos iteraciones, lo cual es sinónimo de menos tiempo de computo, además que disminuye el error, aunque es en una pequeña fracción de 0.25 metros.

Realizando el mismo análisis para el caso del método estático, los resultados indican que los menores valores de error se obtienen utilizando redes neuronales o deep learning, con un error promedio de 4.4857 metros, seguido de 5.0520 metros y finalmente SVM con 7.0241 metros. Las varianzas en este caso son mucho más pequeñas, por lo que los resultados son mucho más estables, lo cual es esperable, ya que no hay movimiento efectivo, por lo que solo afecta el cambio en el entorno y la temporalidad. Por otra parte, el error en cada eje es mucho menor que el método dinámico, lo cual se refleja en el error promedio. A pesar de que el error disminuye, 4 metros aún es demasiado error para posicionamiento exacto, pero se debe recordar que este es un problema de clasificación y para obtener la posición exacta se debe acertar perfectamente en la clasificación y en este caso, la grilla al ser de 4×4 metros, si se falla en la clasificación aunque sea en 1 unidad relativa en la posición, es decir, la clase adyacente, el error ya es de al menos 4 metros, por lo mismo, no es sorpresivo encontrar errores tan altos.

Con respecto al método estático utilizando PCA, nuevamente los errores decrecen en todos los algoritmos, al igual que el error en cada eje y la varianza. Esto demuestra que utilizar PCA es una buena opción si solo se analiza el error promedio. La mejora de las redes neuronales es de 0.55 metros, lo cual es significativo, considerando además que PCA ayuda a reducir el tiempo de procesamiento. El objetivo del método estático es evaluar la estabilidad temporal de los algoritmos, y el que obtiene el mejor resultado es NN, esto se debe a que las redes neuronales no son afectadas mayormente por los cambios del entorno, es decir, el ruido, porque puede reconocer estos patrones espaciales y temporales, y al estar estático mucho tiempo, no se ve mayormente influenciado, por lo que logra determinar de mejor manera la clasificación respectiva. Entonces, para el método estático es mejor utilizar NN con PCA.

Ya que fueron analizadas las métricas básicas, se prosigue a analizar el CDF. Como se menciona anteriormente en general los resultados para el método estático y dinámico son muy similares al momento de analizar el CDF como muestran los gráficos 6.1 y 6.2. A modo de síntesis, se debe tener claro que la mayor cantidad de las veces, aproximadamente el 80 %, el error es menor a 10 metros , y el 60 % de las veces es menor a 8 metros para el método dinámico. Porcentajes menores a estos no son relevantes, ya que no es muy frecuente y tampoco indica valores fiables. Con lo anterior en mente, se debe notar que la mejor curva de CDF para el método dinámico es obtenida por el algoritmo KNN, seguido por SVM y finalmente redes neuronales. Claramente, se ve una distorsión en las redes neuronales al utilizar PCA, que afecta el valor del CDF y su respectivo error, por lo demás los gráficos son muy similares en su forma. Lo que se busca en este tipo de graficas de error, es curvas muy empinadas para valores pequeños de error, ya que esto significa que la probabilidad de obtener estos valores pequeños es muy alta, por lo que la confianza en el algoritmo aumenta,

y en este caso todas las formas de las gráficas son diagonales, a excepción de NN con PCA que es mucho más achatada, lo cual indica que para una determinada probabilidad, el error incrementa mucho, lo cual no es bueno para el algoritmo y lo convierte en poco fiable.

Respecto al CDF de 100 %, es decir, lo que siempre ocurrirá para el método dinámico, el error mínimo que se alcanza sin PCA es presentado por KNN con 16.576 metros y al utilizar PCA, este valor disminuye a 16.1554 metros, obteniendo así una mejora de 2.5375 %. Se debe destacar que el error de redes neuronales con PCA disminuye de 20.5677 a 16.1554 metros, esto es 21.4525 %, que es muy alto, lo que significa que NN con PCA a pesar de ser menos estable y tener mayores errores para porcentajes más bajos(es menos fiable), aun así el error mínimo esperable es mucho mejor que el error mínimo esperable sin utilizar PCA, es decir, mucho mejores valores pero con menos estabilidad en estos. Dicho todo lo anterior, KNN es el claro ganador cuando se analiza CDF para el método dinámico, sobre todo considerando PCA, lo cual es idéntico a lo determinado en el apartado anterior de análisis de error medio o RMSE.

Para el análisis de CDF en el caso estático, los resultados son mucho mejores comparados al caso del método dinámico como era esperable, debido a la estabilidad de las ondas emitidas. Aquí se hace un análisis de cada algoritmo debido a que son resultados muy distintos entre ellos. En primer lugar, KNN tiene errores menores o iguales a 8 metros con una probabilidad de 80 % aproximadamente, y de 4 metros con una probabilidad de 60 %. Por otra parte, cuando se aplica PCA los resultados son idénticos, esto incluso se puede verificar con la forma de las curvas como muestra el grafico 6.2. Respecto a SVM, cuando no se aplica PCA, la curva es muy achatada y el error de 15 metros con una probabilidad de 60 %, lo cual es muy malo. Cuando se adopta SVM con PCA este error disminuye significativamente, con un error de 8 metros con probabilidad de 80 % y de aproximadamente 4 metros con probabilidad de 60 %. Finalmente, cuando se analizan las redes neuronales ocurre una anomalía, ya que el error decrece a 2.5 metros con una probabilidad mayor a 80 %, lo que es una mejora muy significativa y relevante. Esto puede deberse a lo que se menciona anteriormente, las propiedades de mantener el ruido fuera y ser estable temporalmente, es decir, no ser muy influenciado por cambios a través del tiempo. Por otra parte, al aplicar PCA a NN, el error aumenta, llegando como los otros algoritmos a aproximadamente 8 metros con probabilidad mayor al 80 %. Con lo anterior, es claro que redes neuronales es el ganador en este análisis, seguido de KNN y SVM. Es necesario recalcar que PCA no es necesario y que afectaría en este caso los resultados, lo cual no es coincidente con lo obtenido en el análisis de error medio, y esto es perfectamente válido, ya que el CDF muestra solo la probabilidad de que el error sea menor a cierto valor, es por ello que ayuda a tomar la decisión de utilizar o no PCA.

Para el análisis del CDF al 100 %, la Tabla 6.6, muestra que KNN no cambia prácticamente, SVM disminuye 16.1544 a 15.1327 metros, disminuyendo así 6.33 %. Finalmente, las redes neuronales pasan de 11.18033 metros sin PCA a 16.1554 metros con este, lo cual supone un aumento de 44.49 %. Los resultados muestran los errores mínimos asociados. En este caso y como se menciona anteriormente, NN con PCA aumenta el error, mientras SVM con kernel Gaussiano mejora un poco, aunque no siempre ocurrirá esto, ya que PCA origina un espacio no correlacionado y SVM con función de radio basal asume que el espacio se distribuye de manera Gaussiana, lo cual no siempre es cierto, y por lo que tener en cuenta esta mejora tanto para el caso estático y dinámico no es fiable, debido a que puede ser producto de una casualidad. Con lo anterior, es claro que para el método estático es mejor utilizar redes neuronales sin PCA basado en el análisis del CDF.

Para el análisis de distribución de los datos, es necesario analizar los datos relativos a los percentiles o cuartiles, y el análisis cualitativo del diagrama de cajas. Con respecto a los datos de la Tabla 6.7, que representan la distribución obtenida en el método dinámico, se observa que la mediana de los errores se encuentra entre los valores 6.0827 metros y 9.2195 metros. Por otra parte, para el primer cuartil, es decir, Q_1 , que es equivalente al 25avo percentil, el máximo valor es 5 metros en SVM y SVM utilizando PCA, y el mínimo lo obtiene redes neuronales con PCA, con 3 metros, lo cual es indicativo que el 25 % menor de los datos están en este rango, por lo que no es buen indicativo para el posicionamiento, debido a que este error es muy grande. Luego, el tercer cuartil, es decir Q_3 , esta entre 9.2195 metros para KNN y KNN PCA, hasta 15.1327 para NN PCA. Esto indica que la distribución del 50 % total de los datos está en un rango intercuartílico ($Q_3 - Q_1$) de 5.0 metros para SVM PCA en el mejor caso, y 12.1327 metros para el peor caso.

Con lo anterior, es claro que el mejor algoritmo según este análisis es KNN PCA, debido a que sus

valores de $Q1$ y $Q3$ son los más bajos y el rango intercuartílico es no es tan grande, por lo que se puede determinar que el valor máximo de la distribución no escapa tanto, en este caso es 16 metros, que se condice con los análisis de CDF. Los peores resultados en la distribución del error lo obtienen las redes neuronales siempre y cuando se analicen solo los datos del Boxplot, ya que si se toman en cuenta los percentiles 90 y 95, SVM es el peor debido a que alcanza los 20 metros en el 95 % inferior de los datos. El peor valor entonces, según el Boxplot es obtenido por NN PCA, ya que el tercer cuartil es muy elevado, además su rango intercuartílico es muy grande, lo que indicaría que el 50 % central de los datos están demasiado dispersos en este rango, y a pesar de que su distribución general es similar a las demás, ya que el mínimo y máximo están en un rango parecido a KNN por ejemplo, la distribución interna es muy disociada, mostrando así falta de agrupación en los datos, provocando de esta forma resultados poco estables.

Con respecto a la Tabla 6.8, en este caso la mediana disminuye considerablemente con respecto al caso dinámico, lo cual es un indicativo de que la distribución de los datos tiende a estar mucho más cercanos a valores de error menor. Luego, el primer cuartil muestra como los datos se agrupan en valores muy cercanos a 2 metros, a excepción de las redes neuronales sin utilizar PCA, la cual es 3.6055 metros. Continuando con el análisis, para el tercer cuartil, el menor valor es obtenido por las redes neuronales sin PCA, esto como se ha mencionado anteriormente en los análisis de las métricas, es una anomalía, ya que como se observa en la tabla, el primer y tercer cuartil coinciden, lo cual se traduce en un rango intercuartílico igual a cero, vale decir, se asegura que al menos el 50 % de los datos corresponden exactamente a este valor de error, 3.6055 metros, inclusive el valor mínimo y máximo coinciden en este valor, con lo cual se puede establecer que la gran mayoría de los datos de error obtenidos por NN determinaron la misma posición, es por ello que el error se repite y la distribución es uniforme, a excepción de los *outlayers* fuera del rango. Posteriormente, la menor distribución es la de NN con PCA, aunque es idéntica a la presentada por KNN, su mediana es mucho más baja y por otra parte al mirar los percentiles 90 de ambas distribuciones, NN con PCA alcanza los 7.2801 metros, mientras que KNN y KNN con PCA obtienen valores sobre los 15 metros, por lo que la diferencia es muy alta. Finalmente, SVM, a pesar de tener una mediana muy baja, su cuartil 3 es muy alto, lo que implica una distribución muy dispersa sobre el 50 % de los datos centrales, lo que no es bueno por ser demasiado variable, por otra parte, su valor máximo es el más alto, con 16.1554 metros. Finalmente, SVM con PCA logra mejores resultados, sin embargo, su mediana y rango intercuartílico son más grandes que en los otros casos, por lo que queda reseñado a la última posición en términos de distribución.

A continuación se analiza la Figura 6.3, para realizar además un análisis cualitativo. Como se observa en el gráfico, para el caso dinámico (diagramas de caja rojos), KNN y KNN con PCA muestran una distribución estable, y con sus valores mínimos y máximos no muy alejados del 50 % de los datos, seguido posteriormente de SVM y SVM PCA, con un rango intercuartílico estable y pequeño. Estos análisis demuestran que, para los dos algoritmos, la técnica PCA no ayuda demasiado a mejorar los resultados. Finalmente, las redes neuronales, a pesar de las métricas obtenidas en análisis anteriores, acá la distribución es peor, y sus valores de error son más altos, sobre todo en redes neuronales con PCA, en donde el rango intercuartílico es muy grande y la mayor parte de los datos están allí, aunque si bien la mediana es baja, la parte superior de los datos posee errores muy altos, por lo que se torna muy poco estable y los errores muy dispersos.

Respecto al método estático, KNN muestra exactamente el mismo comportamiento, solo que esta vez los errores están por debajo del método dinámico, mientras que en este caso es seguido por redes neuronales con un error y una distribución orientada a valores más bajos. Aquí se muestra la anomalía de la red neuronal sin PCA, en donde se obtiene un mismo resultado todo el tiempo, por lo que es muy estable en sus valores, mientras que al utilizar PCA empeoran los resultados por los motivos explicados anteriormente, en donde las redes neuronales, específicamente deep learning es mucho mejor utilizar los datos completos y que sea el mismo algoritmo el que determina las mejores componentes. SVM por su parte, al no utilizar PCA muestra un rango de valores muy altos, por lo que esta información redundante repercute negativamente cuando las pruebas se realizan por mucho tiempo en una posición estática, por lo que en este caso es mejor realizar PCA para determinar las componentes más favorables, ya que al tener información no relevante, SVM puede verse afectado por sus márgenes y restricciones propias, y pequeños cambios en las ondas electromagnéticas, provoca una mala clasificación, y esto se refleja cuando el tiempo de experimentación es largo.

Finalmente, se debe analizar las diferencias en el tiempo de procesamiento, ya que este es un valor

fundamental para tomar una decisión en la elección del mejor algoritmo. Para ello se analiza la Tabla 6.9, la cual muestra las mejoras de cada algoritmo en tiempo de procesamiento. Se debe notar que para obtener aquellos valores se toma el tiempo de procesamiento en cada ejecución y se promedian para obtener los valores finales. La tendencia es sumamente clara, PCA ayuda a disminuir el tiempo de procesamiento significativamente. KNN por su parte, logra un incremento de 8.1361 %, lo cual es relativamente bajo, sin embargo se debe recordar que el dataset en este caso es pequeño, y a medida que este crece, KNN aumenta significativamente su tiempo de computo, ya que itera sobre cada registro del dataset en una búsqueda exhaustiva, por lo que utilizar PCA con KNN es un requisito fundamental para posicionamiento en interiores, ya que a mayor tiempo de procesamiento, más lentas serán las respuestas, por lo que la posición del usuario siempre tendrá un retraso acumulado, lo que distorsiona su posicionamiento en tiempo real. SVM por su parte reduce su tiempo de computo en un 53.0966 % lo cual es una ganancia muy significativa. Esto se debe a que el tiempo de entrenamiento de SVM kernelizado es cuadrático, por lo mismo demora mucho en entrenarse, sin embargo su tiempo de ejecución es lineal según el número de vectores de soporte, y lineal en el número de características, por lo que al reducir las características casi a la mitad(5 componentes), como hace PCA, el tiempo de procesamiento se reduce a la mitad efectivamente.

Por último, las redes neuronales presentan un tiempo de procesamiento muy bajo, del orden de menos de un milisegundo. Esto se debe a la librería de inferencia de Tensorflow para Android, ya que solo se necesita el grafo de la red neuronal profunda y a partir de los inputs es ejecutada de manera óptima, y precisamente esto destaca a este framework, ya que puede embeber estas grandes redes en pequeños sistemas como un teléfono celular, logrando un rendimiento superior. La mejora es de un 24.0867 %, lo cual es bastante, pero si se observan los valores, el cambio no es drástico para el sistema de posicionamiento, además teniendo en consideración los valores de error analizados anteriormente, esta pequeña ganancia en tiempo, afectaría a los resultados de posición, cosa que es fundamental, por lo mismo no es recomendable aplicar PCA en este caso, a costa de perder tiempo de procesamiento.

A modo de síntesis, respecto a todos los análisis realizados en esta sección, lo primero que se debe destacar es que los resultados estáticos son mucho mejores que los resultados dinámicos, sin embargo, el escenario de que el usuario este estático en un punto no es para nada realista, y no se debe basar tanto la elección en este análisis. Primero, los mejores valores de error medio son obtenidos por KNN y NN, en ambos métodos (estático y dinámico), además NN logra el mejor error en el método estático. Luego, según CDF se determina que KNN y SVM presentan los mejores resultados en el método dinámico, sobre todo sus variantes con PCA. En el método estático, ampliamente ganan las redes neuronales sin PCA, seguido de KNN por tener valores de error pequeños más probables. Cuando se analiza la dispersión en los datos, KNN es mucho menos disperso en ambos métodos y sus errores están más centrados en valores bajos, mientras NN presenta mucho mayor dispersión en el método dinámico, pero casi nada en el método estático, sobre todo al no utilizar PCA. Finalmente, respecto al tiempo, SVM es el claro ganador, seguido de las redes neuronales y KNN. Con lo anterior, la mejor elección se basa en todos estos criterios, por lo que el mejor algoritmo es redes neuronales, debido a que el error respecto a KNN no es tan distinto, y a pesar de que su distribución de error en el método dinámico se orienta a valores más altos y menos estables según CDF (Errores altos ocurren más frecuentemente), esto no influye significativamente, considerando que las medianas son muy similares a SVM y KNN. Además, el tiempo de procesamiento es muy pequeño en comparación a otros algoritmos, lo que influye mucho eventualmente en SVM y KNN ya que dependen del número de características, y NN es independiente de utilizar o no PCA, por lo que las características del dataset no afectan su rendimiento de evaluación, solo el tiempo de entrenamiento, que no es considerado en este análisis. Por todos los motivos mencionados es mejor utilizar redes neuronales sin PCA, luego KNN y finalmente SVM con kernel gaussiano, ambos con PCA.

A pesar de que se logra un posicionamiento cercano, con varianzas pequeñas, las distancias establecidas en metros, no sirven para un posicionamiento exacto en tiempo real, debido a los errores presentados, por lo que este tipo de algoritmo no puede indicar efectivamente la posición exacta en todo momento, sobre todo cuando el usuario está en movimiento, pero si sirve como una guía para establecer una región cercana al usuario o área en donde es más probable que se encuentre el usuario en un determinado tiempo.

7 | Conclusión

La localización en interiores es sin duda uno de los problemas más importantes a resolver en el último tiempo, ya que, así como GPS es la tecnología de facto para posicionar en zonas exteriores, es necesario un framework o sistema global de posicionamiento en interiores, con precisión y estimación de error definidas y estandarizadas. Esto es un problema para los experimentadores, ya que las tecnologías actuales capaces de determinar posición son aún demasiado precarias por una parte y también complejas en su implementación.

Ante estas eventualidades, muchos investigadores han tratado de sobrelevar los problemas relativos al posicionamiento en interiores mediante técnicas que puedan minimizar el error debido a los constantes cambios en la organización de los recintos, ya que este es sin duda el mayor problema, el cambio espacial constante en interiores, lo cual repercute negativamente en todos los métodos y sistemas desarrollados, los cuales deben lidiar con estos cambios, y utilizar métodos para atenuar este ruido.

Para mejorar la exactitud, muchos acercamientos se han presentado, especialmente en el área de redes inalámbricas, ya que su implementación es relativamente fácil, además es posible establecer muchos marcos matemáticos de trabajo que dan muchas posibilidades de trabajar con ondas electromagnéticas. Para ello, se han desarrollado técnicas matemáticas con el fin de mejorar la precisión y disminuir el error, entre ellas algunas basadas en estadísticas, métodos de análisis de señales, comportamiento de las ondas electromagnéticas, métodos deterministas, entre otros.

Como el fin es manejar adecuadamente este ruido inherente que ocurre en interiores, es necesario utilizar técnicas capaces de aprender y que no solo se basen en ciertos comportamientos específicos de un determinado momento, ya que un recinto interior puede cambiar constantemente, ya sea por cambios en infraestructura, tránsito de personas, entre otros. Es por ello que en el presente trabajo se ha decidido utilizar métodos de máquinas de aprendizaje, ya que estos permiten aprender a partir de un grupo de ejemplos, los cuales en este caso pueden ser obtenidos a partir de varios días distintos, con el fin de tener un amplio rango de casos posibles, para que luego el algoritmo pueda generar patrones temporales y espaciales, y así mejorar la exactitud del posicionamiento.

Para el desarrollo de la experimentación se utilizan ondas Bluetooth mediante dispositivos Beacons, para medir que tan buenos resultados aportan a la investigación, además de los ya mencionados métodos de aprendizaje automático. Lo primero a tener en cuenta, es que como se demuestra en la propuesta de solución, las señales Bluetooth y en general cualquier onda electromagnética se ve afectada profundamente por cualquier objeto que se interponga, en este caso se ha probado particularmente con una persona, y como es bien sabido, gran parte del cuerpo humano es agua, con lo cual la señal percibida en el receptor decae, por lo que esto afecta negativamente a los algoritmos y métodos matemáticos, ya que la información porta ruido e interferencia.

Con respecto las mediciones obtenidas, es claro que los datos recolectados presentan estructuras no lineales, pero con correlaciones lineales en sus vecindarios, por lo que en general, los mejores algoritmos de máquinas de aprendizaje son aquellos capaces de reconocer estas estructuras. En este trabajo particularmente se prueban muchos métodos de máquinas de aprendizaje, pero los mejores resultan ser KNN, SVM con Kernel Gaussiano y Redes Neuronales, por lo que se menciona anteriormente, es decir, estos son capaces de reconocer estructuras no lineales.

A partir de la experimentación en la fase online, se determina que el mejor algoritmo para el posicionamiento corresponde a Redes Neuronales artificiales, mediante un análisis de CDF, Boxplot y medidas de error medio como RMSE. Además, se debe considerar el uso de Principal components analysis, el cual permite eliminar la correlación lineal y también disminuir la dimensionalidad de los datos.

El mejor resultado para el método dinámico (en movimiento) corresponde a KNN con PCA, con un error medio de 6.6812 metros, seguido de NN con PCA y SVM con PCA. Por otra parte, para el método estático, los resultados cambian y los mejores valores se obtienen en NN con PCA con 3.9341 metros de error, seguido de KNN con PCA y finalmente SVM con PCA.

Según el estudio, el mejor algoritmo, es decir, el más estable resulta ser KNN, ya que se comporta similar tanto en el método dinámico como estático. Luego redes neuronales presentan los mejores valores de error en términos generales, pero es mucho más inestable, y sus rangos de dispersión más altos, por lo que, aunque puede presentar menores valores de error, esto no siempre ocurre, y en general oscila entre valores bajos y medios de error.

Con respecto al uso de PCA, es claro que utilizarlo es definitivamente la mejor opción, sobre todo en algoritmos como KNN y SVM, debido a que reduce sustancialmente el tiempo de computo, y además mejora la precisión, disminuyendo el error presente en los algoritmos. A pesar de lo anterior, y de que PCA disminuye el error, al utilizarlo en redes neuronales se puede observar que su rango de dispersión aumenta significativamente, e incluso en el método estático presenta peores valores de error, por lo que para redes neuronales artificiales, aunque eventualmente pueda mejorar el tiempo de computo, es mejor no utilizar PCA, ya que NN ya trae incorporado su propio método de disminuir la dimensionalidad al buscar patrones y tendencias locales, con lo cual no es necesario PCA en este último caso.

Con respecto al tiempo, PCA logra disminuir la complejidad computacional en 8.13 % para KNN, 53.09 % para SVM y 24.08 % para NN. Con todo lo anterior, la mejor elección es redes neuronales artificiales siempre y cuando se esté hablando de error, por otra parte si se toma en consideración la estabilidad de los datos, es decir, su dispersión sobre las medidas de tendencia central, KNN es por lejos el ganador. SVM, es sin duda el peor de los tres, ya que no posee mejores valores de error o menor dispersión en sus datos de error.

Los análisis indican entonces que utilizar Bluetooth con técnicas de máquinas de aprendizaje, pueden reducir el error a unos pocos metros, particularmente el mejor valor encontrado lo obtienen las redes neuronales artificiales con 3.9341 metros, lo cual es relativamente alto si se considera un posicionamiento en tiempo real, preciso y sin grandes errores, como es de esperar. Esto indica entonces que el sistema presentado en este trabajo puede ser la base para estimar la posición asociada a una región o zona geográfica de un recinto interior, pero no para determinar efectivamente la localización en tiempo real.

Claramente no se logra el objetivo de obtener una precisión de unos pocos centímetros como logra hacerlo GPS, sin embargo, este trabajo sirve como base para futuras investigaciones, ya que quedan muchas incógnitas abiertas, principalmente análisis de la distribución y densidad del posicionamiento de los Beacons Bluetooth, es decir, estudios sobre cuantos Beacons utilizar sobre una determinada región. Por otra parte, el análisis de la grilla, ya que en este caso se utiliza una de 4×4 metros, lo cual es muy grande, por lo que los errores en la clasificación aumentan significativamente al errar en clasificar; entonces sería razonable analizar qué tamaño de grilla produce los mejores resultados. Por otro lado, sería interesante analizar que ocurre al utilizar las técnicas presentadas en este trabajo, en conjunto con otros métodos como puede ser localización por magnetismo, fusión de sensores iniciales, o las mismas señales WiFi en conjunto con señales Bluetooth.

Con respecto a el modo de recolección de fingerprints, esto es aún un problema para esta técnica, ya que en espacios reducidos, en la fase offline la recolección es mucho más rápida, sobre todo con grillas de tamaño grande, sin embargo, si el recinto contiene muchos puntos de referencia, la recolección es lenta y requiere mucho esfuerzo, ya que no solo basta con la recolección inicial de fingerprints, sino que cada cierto tiempo se debe actualizar los modelos, debido al cambio espacial en la disposición interior de los objetos. Esto es sin duda uno de los problemas más grandes de Fingerprint y que debe ser resuelto para lograr posicionamiento con buena precisión. Una posible solución es tener dispositivos receptores que actualicen

la base de datos en tiempo real, pero ahí el problema es reentrenar los algoritmos de clasificación sobre la marcha, lo cual es muy complejo.

A pesar de que este trabajo solo se utilizan algoritmos de clasificación supervisados, otra alternativa para la localización a nivel de región es utilizar clasificadores no supervisados, es decir, reconocimiento de patrones sin necesidad de indicar la posición, esto reduce considerablemente el esfuerzo para la recolección de fingerprint, ya que en la fase de entrenamiento no es necesario pasar al algoritmo la etiqueta de cada punto, es decir, la posición, sino que es el mismo algoritmo el que se encargaría de reconocer patrones en las señales, sin importar el ruido inherente, por lo que en este contexto, se puede localizar a nivel de regiones, es decir, habitaciones o niveles. Si bien esto no es tan preciso, en conjunto con algoritmos de clasificación supervisados puede ayudar a dar pistas a los algoritmos de machine learning, para así obtener mejores resultados.

A partir de los estudios realizados, se puede determinar que las máquinas de aprendizaje en conjunto con las señales Bluetooth Low Energy pueden ser un punto de partida para mejorar la precisión y disminuir el error del posicionamiento en interiores, en particular el Deep Learning, con el framework Tensorflow, el cual puede ser portado fácilmente a cualquier dispositivo móvil, con lo que es factible utilizar, además de ser sumamente rápido una vez que las redes están entrenadas. Como conclusión final, se debe notar que en este caso solo se utiliza un lugar de experimentación, en particular, un estacionamiento, con lo que estos modelos entrenados no funcionarían en otros recintos. Este es el mayor problema del posicionamiento en interiores, lograr un modelo estándar, que funcione relativamente bien en gran parte de los escenarios y no dependa específicamente del lugar de experimentación. Por el momento esto no es posible, y sigue en constante investigación, ya que aún se deben resolver muchas preguntas que están abiertas para lograr realmente un sistema fiable y preciso que pueda localizar en interiores.



Bibliografía

Abadi, Martín; Agarwal, Ashish; Barham, Paul; Brevdo, Eugene; Chen, Zhifeng; Citro, Craig; Corrado, Greg S.; Davis, Andy; Dean, Jeffrey; Devin, Matthieu; Ghemawat, Sanjay; Goodfellow, Ian; Harp, Andrew; Irving, Geoffrey; Isard, Michael; Jia, Yangqing; Jozefowicz, Rafal; Kaiser, Lukasz; Kudlur, Manjunath; Levenberg, Josh; Mané, Dan; Monga, Rajat; Moore, Sherry; Murray, Derek; Olah, Chris; Schuster, Mike; Shlens, Jonathon; Steiner, Benoit; Sutskever, Ilya; Talwar, Kunal; Tucker, Paul; Vanhoucke, Vincent; Vasudevan, Vijay; Viégas, Fernanda; Vinyals, Oriol; Warden, Pete; Wattenberg, Martin; Wicke, Martin; Yu, Yuan; y Zheng, Xiaoqiang (2015). TensorFlow: Large-scale machine learning on heterogeneous systems. Software available from tensorflow.org. 5.3.3

Agnieszka Gasiorek (2014). <https://kontakt.io/blog/beacon-configuration-strategy-guide-interval/>. [Acceso Enero 2018]. 5.2

Anand, U. Viney (2014). Smart indoor localization using machine learning techniques. In *SMART INDOOR LOCALIZATION USING MACHINE LEARNING TECHNIQUES* (pp. 1223–1228). 5.9

Bahl, Paramvir y Padmanabhan, Venkata N. (2000a). Radar: an in-building rf-based user location and tracking system. In *RADAR: an in-building RF-based user location and tracking system* (pp. 775–784). 1

Bahl, P. y Padmanabhan, V. N. (2000b). Radar: an in-building rf-based user location and tracking system. In *Proceedings IEEE INFOCOM 2000. Conference on Computer Communications. Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies (Cat. No.00CH37064)*, volume 2 (pp. 775–784 vol.2). 1

c. Zufferey, J.; Klaptocz, A.; Beyeler, A.; d. Nicoud, J.; y Floreano, D. (2006). A 10-gram microflyer for vision-based indoor navigation. In *2006 IEEE/RSJ International Conference on Intelligent Robots and Systems* (pp. 6–6). 3.1.1, 3.1, 3.2

Chan, Li-wei; Chiang, Ji-rung; Chen, Yi-chao; Ke, Chia-nan; Hsu, Jane; y Chu, Hao-hua (2006). *Collaborative Localization: Enhancing WiFi-Based Position Estimation with Neighborhood Links in Clusters*, (pp. 50–66). Springer Berlin Heidelberg: Berlin, Heidelberg. 2

Chen, L. H.; Wu, E. H. K.; Jin, M. H.; y Chen, G. H. (2014). Intelligent fusion of wi-fi and inertial sensor-based positioning systems for indoor pedestrian navigation. *IEEE Sensors Journal*, 14(11), 4034–4042. 3.1.4

Estimote (2017). <https://estimote.com/>. [Acceso Noviembre 2017]. 4.1.3

Fang, B. T. (1990). Simple solutions for hyperbolic and related position fixes. *IEEE Transactions on Aerospace and Electronic Systems*, 26(5), 748–753. 1a

Fang, S. H.; Wang, C. H.; Huang, T. Y.; Yang, C. H.; y Chen, Y. S. (2012). An enhanced zigbee indoor positioning system with an ensemble approach. *IEEE Communications Letters*, 16(4), 564–567. 3.1.7

Farshad, A.; Li, Jiwei; Marina, M. K.; y Garcia, F. J. (2013). A microscopic look at wifi fingerprinting for indoor mobile phone localization in diverse environments. In *International Conference on Indoor Positioning and Indoor Navigation* (pp. 1–10). 1

- Florian Trautmann (2015). <https://kontakt.io/blog/estimote-vs-kontakt-io-practical-tests/>. [Acceso Enero 2018]. 4.1.3
- Haeberlen, Andreas; Flannery, Eliot; Ladd, Andrew M.; Rudys, Algis; Wallach, Dan S.; y Kavraki, Lydia E. (2004). Practical robust localization over large-scale 802.11 wireless networks. In *Proceedings of the 10th Annual International Conference on Mobile Computing and Networking*, MobiCom '04 (pp. 70–84). New York, NY, USA: ACM. 2
- Harle, R. (2013). A survey of indoor inertial positioning systems for pedestrians. *IEEE Communications Surveys Tutorials*, 15(3), 1281–1293. 3
- He, S. y Chan, S. H. G. (2016). Wi-fi fingerprint-based indoor positioning: Recent advances and comparisons. *IEEE Communications Surveys Tutorials*, 18(1), 466–490. 3.11
- Honkavirta, V.; Perala, T.; Ali-Loytty, S.; y Piche, R. (2009). A comparative survey of wlan location fingerprinting methods. In *2009 6th Workshop on Positioning, Navigation and Communication* (pp. 243–251). 2
- Khalajmehrabadi, A.; Gatsis, N.; y Akopian, D. (2017). Modern wlan fingerprinting indoor positioning methods and deployment challenges. *IEEE Communications Surveys Tutorials*, 19(3), 1974–2002. 3.2.3.1
- Kontakt (2017). <https://kontakt.io/use-cases/>. [Acceso Noviembre 2017]. 3.1.6, 4.1.3
- Ladd, Andrew M.; Bekris, Kostas E.; Rudys, Algis; Marceau, Guillaume; Kavraki, Lydia E.; y Wallach, Dan S. (2002). Robotics-based location sensing using wireless ethernet. In *Proceedings of the 8th Annual International Conference on Mobile Computing and Networking*, MobiCom '02 (pp. 227–238). New York, NY, USA: ACM. 2
- Link, J. B.; Smith, P.; Viol, N.; y Wehrle, K. (2011). Footpath: Accurate map-based indoor navigation using smartphones. In *Indoor Positioning and Indoor Navigation (IPIN), 2011 International Conference on* (pp. 1–8). 3.1.7, 3.4
- Liu, Hui; Darabi, H.; Banerjee, P.; y Liu, Jing (2007). Survey of wireless indoor positioning techniques and systems. *Trans. Sys. Man Cyber Part C*, 37(6), 1067–1080. 3.5, 3.2.1, 3.6, 3.7, 3.8, 3.9, 1, 3.10
- Maribel Tirados (2014). <http://www.bigdatahispano.org/noticias/una-invitacion-al-aprendizaje-automatico/>. [Acceso Enero 2018]. 4.4
- Mautz, R. y Tilch, S. (2011). Survey of optical indoor positioning systems. In *2011 International Conference on Indoor Positioning and Indoor Navigation* (pp. 1–7). 3.1.1
- Nashville GOB (2017). Mayor, music city center unveil wayfinding app. <http://www.nashville.gov/News-Media/News-Article/ID/3477/Mayor-Music-City-Center-Unveil-Wayfinding-App>. [Acceso Noviembre 2017]. 3.1.6
- Ni, L. M.; Liu, Yunhao; Lau, Yiu Cho; y Patil, A. P. (2003). Landmarc: indoor location sensing using active rfid. In *Pervasive Computing and Communications, 2003. (PerCom 2003). Proceedings of the First IEEE International Conference on* (pp. 407–415). 1
- Pedregosa, F.; Varoquaux, G.; Gramfort, A.; Michel, V.; Thirion, B.; Grisel, O.; Blondel, M.; Prettenhofer, P.; Weiss, R.; Dubourg, V.; Vanderplas, J.; Passos, A.; Cournapeau, D.; Brucher, M.; Perrot, M.; y Duchesnay, E. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12, 2825–2830. 5.3.2
- Ramananda Shetty (2015). <https://www.linkedin.com/pulse/eddystone-googles-beacon-profile-ramananda-shetty>. [Acceso Enero 2018]. 4.1, 4.2
- Rüppel, U.; Stübbe, K. Marcus; y Zwinger, U. (2010). Indoor navigation integration platform for firefighting purposes. In *Indoor Positioning and Indoor Navigation (IPIN), 2010 International Conference on* (pp. 1–6). 2.1

- Salamah, A. H.; Tamazin, M.; Sharkas, M. A.; y Khedr, M. (2016). An enhanced wifi indoor localization system based on machine learning. In *2016 International Conference on Indoor Positioning and Indoor Navigation (IPIN)* (pp. 1–8). 4.2.1
- Sun, W.; Liu, J.; Wu, C.; Yang, Z.; Zhang, X.; y Liu, Y. (2013). Moloc: On distinguishing fingerprint twins. In *2013 IEEE 33rd International Conference on Distributed Computing Systems* (pp. 226–235). 1
- Syed Danish Ali (2016). <https://www.analyticsvidhya.com/blog/2016/08/evolution-core-concepts-deep-learning-neural-networks/>. [Acceso Enero 2018]. 4.7, 4.8
- Tarrio, P.; Cesana, M.; Tagliasacchi, M.; Redondi, A.; Borsani, L.; y Casar, J. R. (2011). An energy-efficient strategy for combined rss-pdr indoor localization. In *Pervasive Computing and Communications Workshops (PERCOM Workshops), 2011 IEEE International Conference on* (pp. 619–624). 3.1.7, 3.4
- Tarzia, Stephen P.; Dinda, Peter A.; Dick, Robert P.; y Memik, Gokhan (2011). Indoor localization without infrastructure using the acoustic background spectrum. In *Proceedings of the 9th International Conference on Mobile Systems, Applications, and Services, MobiSys '11* (pp. 155–168). New York, NY, USA: ACM. 3.1.3, 3.3
- Torrieri, D. J. (1984). Statistical theory of passive location systems. *IEEE Transactions on Aerospace and Electronic Systems*, AES-20(2), 183–198. 1b
- Veen, B. D. Van y Buckley, K. M. (1988). Beamforming: a versatile approach to spatial filtering. *IEEE ASSP Magazine*, 5(2), 4–24. 2a
- Wang, He; Sen, Souvik; Elgohary, Ahmed; Farid, Moustafa; Youssef, Moustafa; y Choudhury, Romit Roy (2012). No need to war-drive: Unsupervised indoor localization. In *Proceedings of the 10th International Conference on Mobile Systems, Applications, and Services, MobiSys '12* (pp. 197–210). New York, NY, USA: ACM. 2
- Xiao, Z.; Wen, H.; Markham, A.; y Trigoni, N. (2014). Lightweight map matching for indoor localisation using conditional random fields. In *IPSN-14 Proceedings of the 13th International Symposium on Information Processing in Sensor Networks* (pp. 131–142). 3
- Yeh, L. W.; Hsu, M. S.; Lee, Y. F.; y Tseng, Y. C. (2009). Indoor localization: Automatically constructing today's radio map by irobot and rfids. In *Sensors, 2009 IEEE* (pp. 1463–1466). 1
- Zhou, Junyang; Chu, K. M. K.; y Ng, J. K. Y. (2005). Providing location services within a radio cellular network using ellipse propagation model. In *19th International Conference on Advanced Information Networking and Applications (AINA'05) Volume 1 (AINA papers)*, volume 1 (pp. 559–564 vol.1). 1c