ESCUELA TÉCNICA SUPERIOR DE INGENIEROS INDUSTRIALES Y DE TELECOMUNICACIÓN

UNIVERSIDAD DE CANTABRIA

Trabajo Fin de Grado

Optimización de algoritmos para la estimación de niveles de llenado de contenedores
(Dumpster filling level estimation algorithm optimization)

Para acceder al Título de

Graduado en
Ingeniería de Tecnologías de Telecomunicación

Autor: Jesús Ruiz Bolado

Octubre - 2015

GRADUADO EN INGENIERÍA DE TECNOLOGÍAS DE TELECOMUNICACIÓN

CALIFICACIÓN DEL TRABAJO FIN DE GRADO

Realizado por: Jesús Ruiz Bolado
Director del TFG: Jesús María Ibáñez Díaz
Título: "Optimización de algoritmos para la estimación de niveles de llenado de contenedores"
Title: "Dumpster filling level estimation algorithm optimization"

Presentado a examen el día:

para acceder al Título de

GRADUADO EN INGENIERÍA DE TECNOLOGÍAS DE TELECOMUNICACIÓN

Composición del Tribunal:
Presidente (Apellidos, Nombre): Javier Vía Rodríguez
Secretario (Apellidos, Nombre): Félix Fanjul Vélez
Vocal (Apellidos, Nombre): Jesús María Ibáñez Díaz

Este Tribunal ha resuelto otorgar la calificación de: .....

Fdo.: El Presidente

Fdo.: El Secretario

Fdo.: El Vocal

Fdo.: El Director del TFG
(sólo si es distinto del Secretario)

Vº Bº del Subdirector

Trabajo Fin de Grado Nº
(a asignar por Secretaría)

Agradecimientos

Sería tremendamente errado no hacer un apartado para recordar a todas aquellas personas que han permitido que este trabajo haya podido realizarse sin que nadie se lleve las manos a la cabeza al leerlo.

Primeramente debería dar gracias a aquel que me ha estado supervisando este trabajo desde su comienzo, y al que le debo mucho, ya sea en tiempo de resolver dudas como de conocimientos a lo largo de la carrera. Así que, Jesús Ibáñez, gracias por aguantar todas las preguntas que te he hecho, resolviendo cada una de ellas con paciencia y dedicación. Tampoco debería olvidarme de todos aquellos profesores de la Universidad de Cantabria que me han proporcionado los conocimientos necesarios para poder realizar este trabajo, haciendo hincapié en aquellos pertenecientes al Grupo de Tratamiento Avanzado de Señal, que consiguieron que me interesara por ese inmenso mundo que es el procesado de señal. Mención especial a Javier Vía, que ha aportado bastante a este trabajo, sobre todo en la parte de optimización convexa.

Por otro lado, me gustaría agradecer a mi madre por todo el apoyo que me ha ofrecido y lo, a veces demasiado, bien que cuida de mí. A mi padre por la ayuda que me ha dado a la hora de redactar y a su ojo para ver errores, y a mi hermana, simplemente por ser como es y animarme la existencia.

A mis amigos también debería darles un pequeño apartado, tanto por aguantarme y entender que haya estado largas temporadas viéndoles muy poco, como por animar las noches y hacer de Santander un lugar entretenido. Antonio, Pelayo, Rodri, Xisco… y todos los demás, gracias por estar ahí. Por supuesto, no me puedo olvidar de ti, Marina, que has formado parte de este trabajo apuntando errores y mejorando la corrección de la redacción, así como apoyándome y dándome ánimos cuando estaba bajo de pilas. Y no puedo negar la ayuda de compañeros de la carrera como Alberto, Arturo, Adrián, Julián o Laura, que han estado siempre ahí, luchando por avanzar año a año.

A todos vosotros, os doy las gracias, y espero que algún día sea capaz de devolveros todo lo que me habéis aportado a mí.

Índice general

Palabras clave

Redes de sensores, ultrasonidos, optimización convexa, aprendizaje máquina, máquina de vectores soporte

Keywords

Sensor networks, ultrasounds, convex optimization, machine learning, support vector machines

1. Introducción. Motivación y Objetivos

Hoy en día, las redes de sensores son cada vez más comunes en la vida diaria. Su capacidad de aportar datos en tiempo real las hace muy valiosa en ámbitos tan variados como seguridad, domótica, medicina, aplicaciones industriales…. Uno de estos ámbitos son las *Smart Cities*, en la medida en que las redes de sensores permiten la obtención de datos del tráfico, controlar el alumbrado urbano, llevar el recuento de las plazas de aparcamiento o, algo que ha cobrado interés últimamente, gestionar la recogida de residuos controlando el nivel de llenado de los contenedores. En este aspecto, que es el que nos ocupa, las razones de su uso no son pocas. El poder disponer de información sobre el grado de llenado de los contenedores en tiempo real permite optimizar las rutas de recogida de residuos, reduciendo tiempos y evitando viajes innecesarios. Otro punto importante es que, al evitar que los cubos estén completamente llenos, se evita la acumulación de residuos a su alrededor, lo que redunda en beneficio tanto de los trabajadores encargados de la recogida como de los vecinos; por esto, es necesario disponer de una red de sensores que permita conocer los niveles de llenado de cada contenedor. Sin embargo, diversos problemas, como la irregularidad de los residuos o las variaciones de temperatura, pueden provocar dificultades a la hora de dar con el nivel de llenado; de esta manera, es preciso realizar un procesado de los datos que corrija estos y otros problemas que puedan surgir.

Es en este punto donde se comienza el presente trabajo de fin de grado, partiendo del despliegue de sensores por toda una ciudad los cuales hacen uso de unos algoritmos que no son capaces de adaptarse correctamente a los diferentes problemas que se presentan. Por tanto este trabajo pretende encontrar una solución óptima que corrija estas deficiencias, para lo que se deben alcanzar tres puntos. En primer lugar, encontrar una manera justa que permita comparar los algoritmos que había planteados, buscando una métrica que se adapte a las características del problema y que compare objetivamente los distintos algoritmos. Seguidamente, a partir de este criterio de comparación, se debe decidir cuál es el algoritmo que mejores prestaciones ofrece en base a las limitaciones que este problema impone. Una vez obtenidos estos dos puntos, el último paso es la optimización del algoritmo seleccionado, investigando que herramientas se podrían

utilizar y usándolas para elaborar un método que permita obtener los parámetros óptimos que proporcionen los mejores resultados.

De esta forma, en el capítulo 2 se hará una presentación del problema, explicando los principales elementos con los que se trabajará, así como las trabas anteriormente mencionadas y con las que se debe lidiar. En el capítulo 3, se describirán las soluciones que había planteadas, desarrollando los diversos algoritmos heurísticos que estaban implementados. En paralelo, se estudiará la viabilidad de utilizar algoritmos de aprendizaje supervisado, además de abordarse el estudio de diversas métricas para buscar un criterio justo de comparación. También se incluyen los análisis que se realizaron para decidir qué algoritmo era el más idóneo para este problema y las conclusiones derivadas de estos. Como continuación directa, en el capítulo 4 se buscará una manera de optimizar el algoritmo elegido y los cálculos y análisis que se realizaron en el proceso, terminando con los resultados de la optimización. Finalmente, en el capítulo 5 se desarrollarán las conclusiones finales del trabajo, valorando los resultados obtenidos y haciendo hincapié en las ventajas que puede ofrecer, así como las líneas futuras que se podrían plantear.

2. Caracterización del problema

En este capítulo se procederá a explicar la situación de partida a la hora de enfrentarse al problema a tratar. Por tanto, se explicará el objetivo que se quería conseguir y los elementos con los que se trabajará, así como los problemas que surgieron.

El problema, a simple vista, es sencillo. Se busca poder obtener, de forma remota, los niveles de llenado de los contenedores de papel y de envases de una ciudad mediante el uso de una red de sensores. De esta forma se pretende hacer más eficiente la recogida de deshechos, evitando, por ejemplo, que los trabajadores de la empresa adjudicataria deban realizar desplazamientos innecesarios, ya que disponen de información actualizada sobre qué contenedores están llenos. Esto permitiría diseñar, en tiempo real, una ruta para los camiones recogedores que minimizase las distancias y los tiempos. Por otra parte disponer de esta herramienta redundaría en beneficio de la comunidad y facilitaría los trabajos de recogida ya que no se acumularía basura en la calle dado que el contenedor se habría limpiado antes de llegar al límite de su capacidad. A continuación se habla sobre los contenedores y los sensores que se utilizan en este problema.

2.1 Los contenedores

Los contendores más habituales en las ciudades, y con los que se trabajará, son los denominados contenedores de carga lateral, siendo un ejemplo los fabricados por la empresa RosRoca. Como ya se ha comentado, solo son de papel y de envases, y sus dimensiones son iguales para ambos tipos. En la figura 1 se puede apreciar la forma y dimensiones básicas, siendo la altura (E en el diagrama) la más importante de todas ellas. Su valor es de 1,755m, y será el parámetro a utilizar para calcular el llenado del cubo.

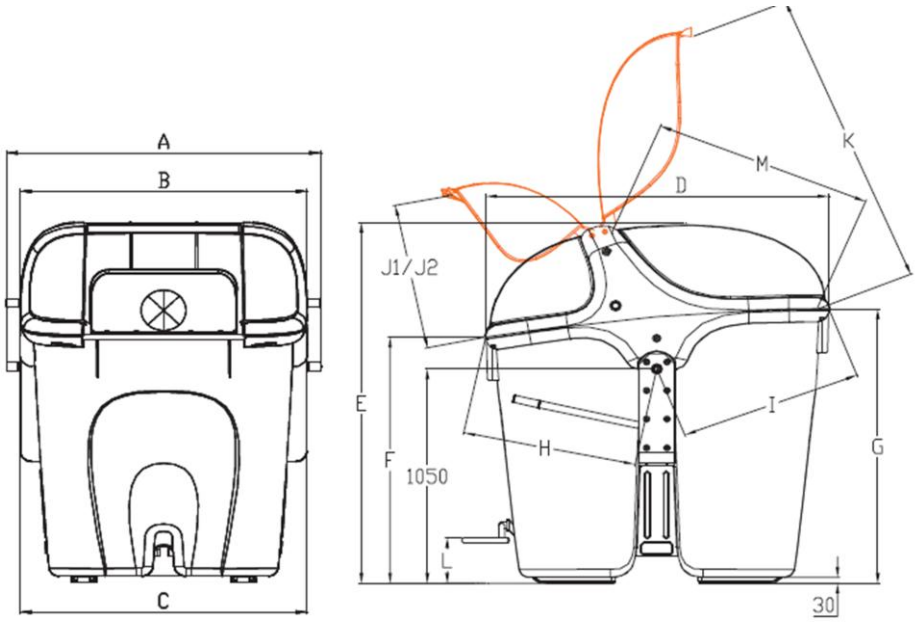

Figura 1. Diagrama de los contenedores

2.2 Los sensores

En cuanto a los sensores que se eligieron, se buscaba principalmente que fueran de bajo coste, ya que debe tenerse en cuenta que el proyecto necesitaba hacer uso de un alto número de dispositivos (una pareja por contenedor) los cuales se desplegarían por toda una ciudad, elevando en gran medida el coste total. Por otro lado, también se buscaba que fueran resistentes, pues iban a trabajar en condiciones ambientales muy exigentes como agua a presión durante lavado de los cubos, o golpes derivados de su recogida. Por último, debían tener una batería de duración suficiente para que fueran autónomos durante un largo periodo de tiempo, evitando el gasto que supondría tener que cambiar los sensores frecuentemente. En la figura 2 se muestra el diagrama de los sensores elegidos.

En cuanto a la tecnología, en base a las características que se buscaban, se había decidido que la que más se adecuaba a ellas y al problema era la de ultrasonidos. Esta tecnología consiste, en pocas palabras, en la emisión de uno o varios pulsos hacia un obstáculo, los cuales al chocar contra él rebotan. Estos rebotes son captados por el sensor y en base al tiempo que hayan tardado en volver se puede dilucidar la distancia a la que se encuentra el obstáculo, pues la velocidad del sonido en el medio de trabajo es conocida. El bajo coste de esta tecnología respecto a otras, como por ejemplo el láser, la

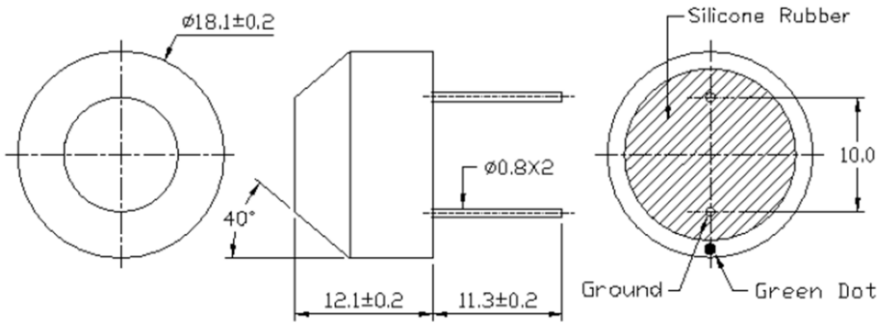

Figura 2. Diagrama de los sensores. Las distancias están en mm

hacen idónea en este campo, y su bajo consumo permite dar mayor autonomía a los sensores, lo que a la larga, como ya se ha comentado, redunda en un mayor ahorro. A cambio las prestaciones que ofrecen los ultrasonidos quedan reducidas frente a las que se podrían conseguir mediante otras tecnologías. Esto obliga a realizar un procesado adecuado de la señal recibida para poder dilucidar correctamente el nivel de llenado, y es esto lo que este trabajo pretende conseguir.

Estos sensores, que están dispuestos en la tapa de los contenedores en parejas de transmisor y receptor, están programados para que manden un tren de pulsos que rebote en la basura y sea recibido de vuelta. Una vez adquirida, la señal se muestrea, obteniéndose 500 muestras. Después de un pequeño procesado, en el que se incluye el filtrado mediante el filtro adaptado, al final del proceso se tiene una señal banda base de 492 muestras.

2.3 Ventaneado de la señal

Debido a que no se buscaba una gran precisión en cuanto a distancia, y que enviar 492 muestras cada vez que el sensor se comunicara con el punto de acceso sería ineficiente tanto para el tráfico de la red como para la batería del sensor, se usan 15 ventanas, que serán indicadores relativos de cómo de lleno se encuentra el cubo. De esta manera, solo se envían 15 datos en vez de 492, lo que reduce el gasto de energía drásticamente, y evita saturar la red. Para calcular los valores de estas ventanas se recurrió a la siguiente manera. Sabiendo que la señal acústica, tras ser muestreada se compone de 492

muestras, es lógico pensar que cada ventana comprenda $\frac{492}{15} \approx 33$ muestras[1]. A partir de estas muestras se calcula la potencia instantánea de cada una de ellas, y se suman mediante la ecuación

$$w_k = \sum_{n=33(k-1)}^{33k} x^2[n], \qquad k \in 1,2,3,\dots,15. \tag{2.1}$$

siendo $x^2[n]$ el valor de la potencia instantánea de la muestra n.

De esta manera el valor de cada ventana no es más que la potencia total en un intervalo temporal concreto. No obstante, para tener una noción completa de la información que aporta una ventana es útil conocer que distancia comprende cada una de ellas y así poder hacer una estimación más visual de como de lleno está cada contenedor. Por tanto, para poder saber cuántos centímetros recorre la señal en cada muestra se utiliza la expresión

$$d = \frac{v_s n}{2 f_s}, \tag{2.2}$$

donde v_s es la velocidad del sonido, n el número de muestra y f_s es la frecuencia de muestreo del sensor (50kHz). El número dos que aparece en el denominador se explica porque la señal va y viene. Sustituyendo los valores en la fórmula se obtiene que la distancia que avanza la señal en cada muestra es $d_0 = 0.343$ cm/muestra. Por tanto, si cada ventana comprende 33 muestras la distancia que comprende cada ventana es $d_{ventana} = 33 * d_0 = 11.32$ cm. Así, la ventana 1 indicaría que el cubo está lleno hasta al menos unos 159 cm desde el suelo, y la ventana 15 que se encuentra vacío, o con un nivel de residuos de unos 11 cm desde el suelo. Como ejemplo gráfico, en la figura 3 se representan las señales correspondientes a una situación en las que el obstáculo se encuentra a unos 115 cm del sensor.

[1] En realidad, debido a que 492 no es divisible entre 15, lo que se hace es asignar a cada ventana 33 muestras, dejando a la última con 31. Para simplificar se supondrá a lo largo del trabajo que todas se componen de 33 muestras.

Se debe hacer mención a la señal que se ve al inicio en las tres figuras, la cual se corresponde a un fenómeno que a lo largo del trabajo denominaremos *crosstalk*. Este *crosstalk* es un problema derivado de los sensores y otras causas que produce un nivel de amplitud al inicio de todas las señales. Más adelante se hablará de él en profundidad. Siguiendo con las figuras, en la 3.a se puede apreciar el pulso de vuelta entre las muestras 300 y 375, que siguiendo la ecuación (2.2) se corresponde con una distancia de entre 103 y 128 cm. La siguiente figura, la 3.b, es la potencia instantánea de la señal

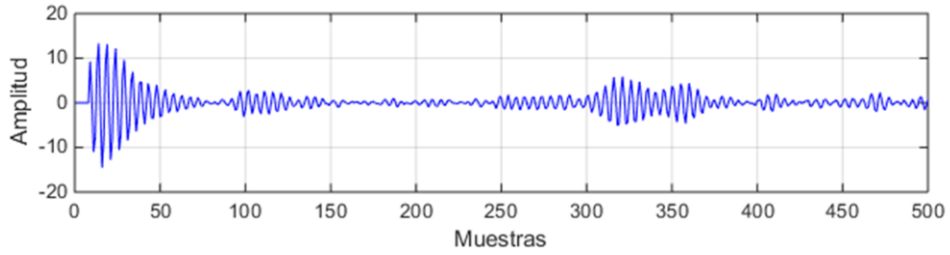

Figura 3.a Ejemplo de la señal muestreada

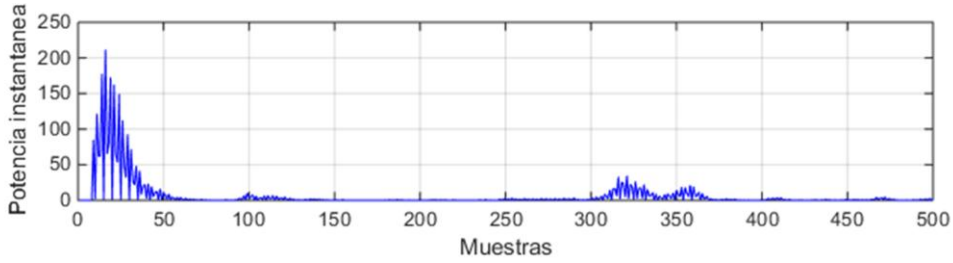

Figura 3.b Ejemplo de potencia instantánea de la señal

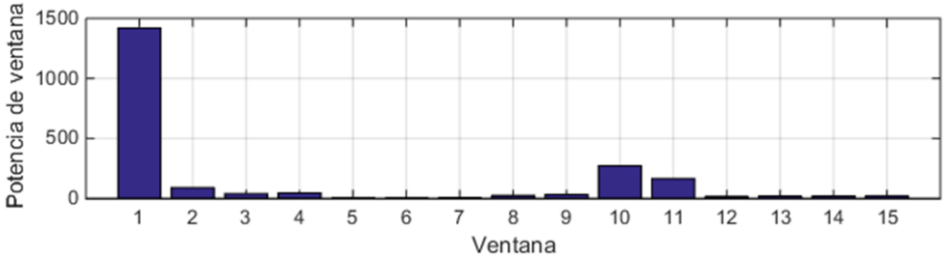

Figura 3.c Ejemplo de la señal tras haber sido "ventaneada"

ejemplo. Esta potencia es la que se sumara en bloques de 33 para calcular los valores de cada ventana, que es lo que se muestra en la figura 3.c. Se puede observar como las ventanas 10 y 11 se corresponden con la posición de los pulsos que se veían en la figura 3.a, siendo más alta la ventana 10 debida a su cercanía y a que el obstáculo se encuentra en su mayoría en espacio correspondiente a esta ventana. También ocupa parte de lo que sería la ventana 11, lo que explica ese pequeño pico.

Otros datos que el sensor permite recabar son la temperatura y la hora, además de una estima de la varianza denominada sigma2, que permite poder determinar si la captura de datos se vio influenciada por un gran ruido ambiente, por ejemplo, unas obras cercanas o la lluvia golpeando la tapa del contendedor. Unos ruidos tan fuertes (para el sensor) podrían dejar la captura inservible, y es mediante esta variable como se puede deducir la existencia de este factor de alteración. Por tanto los datos que se recaban del sensor son los valores de las quince ventanas, la temperatura, sigma2 y el nivel de batería, además el valor del nivel de llenado.

2.4 Problemas encontrados

Sin embargo se habían observado ciertos problemas derivados por una parte de la física del problema y por otra del comportamiento de los sensores utilizados. Son estos los problemas que se deben de afrontar para poder encontrar una solución viable.

2.4.1 Variaciones de la ganancia del sistema

Por un lado se encuentran las altas variaciones en la energía de la señal que se producen en los sensores, debidas en ocasiones a los cambios de temperatura. Si solo se debieran a esto, el problema se podría intentar solucionar encontrando alguna relación entre las variaciones y la temperatura del contenedor, ya que es un valor que se conoce gracias al sensor. Pero también se observaron grandes variaciones en la energía del mismo sensor a temperaturas similares, lo que imposibilitaba el uso de unos umbrales dinámicos. Las razones de estas variaciones siguen sin ser conocidas totalmente a día de hoy, pero se cree que las principales razones son la temperatura y la presión atmosférica. Por otro lado, entre los propios sensores se vio una gran variabilidad a la hora de tomar medidas en las mismas condiciones, lo que dificulta más aun la búsqueda de una solución universal que determine la ventana de llenado.

2.4.2 Crosstalk

Por otro lado, se encuentra el *crosstalk*. Este fenómeno se debe a que, al encontrarse el transmisor y el receptor tan cerca, parte de la señal se introduce desde el primero al segundo. En este caso concreto, también se presume que las vibraciones de la pieza en el que están instalados transmisor y receptor contribuyen a incrementar este efecto. Todo esto produce que en las primeras muestras de la señal aparezca una cantidad de

potencia tremendamente alta independientemente del contenido, lo que se traduce en unos valores muy elevados para las ventanas 1 y 2, como se puede observar en la figura 4. Este fenómeno complica bastante el problema, pues se debe trabajar con unos valores de potencia muy altos en las primeras ventanas, lo que hace que el método que se use para la decisión lo deba tener en cuenta. Sin embargo, también se observó que cuando había un obstáculo muy cercano (en el rango de las ventanas 1 o 2) la potencia de las ventanas aumentaba muy rápidamente, lo cual hace más sencillo lidiar con este problema al permitir diferenciar fácilmente entre un obstáculo en la ventana 1 y/o 2 y el *crosstalk* intrínseco en ella.

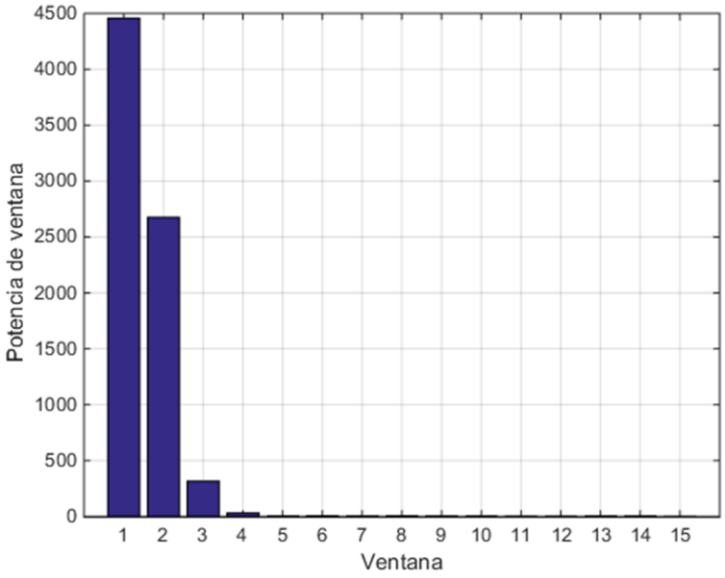

Figura 4. Ejemplo de *crosstalk*. Este ejemplo corresponde a un contenedor completamente vacío, sin embargo se aprecian claramente los altos niveles de potencia en las dos primeras ventanas debidos al *crosstalk*

2.4.3 Rebotes cercanos

En muchos sensores se encontraron pequeños picos de potencia en las ventanas 4 y 5, independientemente del llenado del contendor. Aunque no se está completamente seguro se cree que se deben a rebotes de la señal acústica en la tapa del cubo. El hecho de que en algunos sensores aparezca y en otros no hace más complicada aun la búsqueda de una solución genérica válida para todos los casos.

2.4.4 Variabilidad del residuo

En este problema, se ha de tener en cuenta que la geometría del residuo es tremendamente variable, pudiendo ser tanto plana para una caja de cartón, como irregular para una bola de papel. Asimismo, se pueden producir grandes diferencias en la señal recibida únicamente variando la orientación de estos obstáculos. Y a todo esto hay que añadir que la disposición de dichos residuos es altamente irregular, lo que forma espacios vacíos, montículos y pendientes dentro del contenedor. La unión de todos estos factores produce grandes diferencias entre dos señales para un mismo nivel de llenado, apareciendo niveles de potencia muy bajos debido a que la superficie que presenta el obstáculo es irregular o de pequeñas dimensiones, u obteniendo en el receptor varios picos de potencia debido a rebotes que llegan con suficiente fuerza. Además hay que tener en cuenta que el coeficiente de reflexión varía en función del residuo (un cartón no tiene el mismo coeficiente de reflexión que una bolsa de plástico, o una lata), lo que hace que algunas señales lleguen al receptor mucho más atenuadas que otras. Esto aporta aún más diversidad a las señales recibidas y añade una complicación adicional al problema.

2.4.5 Ruido

Por último, aunque no es una complicación específica de este problema, se debe tener en cuenta el ruido térmico. Este problema puede llegar a ser bastante grave, ya que en muchas ocasiones el nivel de señal que llega al receptor es tan bajo que apenas destaca respecto al nivel de ruido, concretamente en las últimas ventanas. Por otro lado, las interferencias debidas a ruidos externos, por ejemplo obras cercanas, pueden provocar graves errores, aunque para ello se calcula una estima de la varianza (sigma2) que evita en gran medida esta traba. Por lo tanto, las operaciones que hagan los algoritmos deben tener en cuenta estos fenómenos o puede que se produzcan errores en la determinación del nivel de llenado.

2.5 Algoritmo inicial

En principio, para llevar a cabo la decisión del nivel de llenado del contenedor, se había recurrido a un algoritmo muy sencillo al que se denomina algoritmo de comparación de umbrales. Como su nombre indica, basa su funcionamiento en la comparación del valor de las ventanas con otro valor, llamado umbral, distinto para cada una de las ventanas. Sin embargo, se había observado que para este problema las prestaciones de este algoritmo no eran muy buenas, cometiendo errores en múltiples ocasiones debido principalmente a que no tenía en cuenta ninguno de los problemas antes mencionados (a excepción del *crosstalk*). Dos algoritmos nuevos, denominados algoritmo de compensación de atenuación y algoritmo diferencial, también estaban planteados, con los que se buscaba mejorar las prestaciones que ofrece el algoritmo de umbrales.

En el siguiente capítulo se explicarán más detenidamente estos algoritmos, y se ahondará en cómo afrontar el problema de buscar el nivel de llenado de los cubos y cuál sería la solución óptima.

3. Alternativas de solución al problema

En el capítulo anterior se ha tratado el problema original tal y como se encontró al comienzo de este trabajo, explicando sus principales características además de los diferentes obstáculos que presenta y que se deben superar. En este capítulo se extenderán estas nociones, detallando el funcionamiento, además de las ventajas y desventajas de los algoritmos que se han comentado anteriormente. Por otro lado, también se hablará de la idea del uso de máquinas de vectores soporte (SVMs: *Support Vector machines*) que se planteó como una posible solución alternativa.

3.1 Algoritmos de aprendizaje supervisado

En un inicio solo se encontraba la idea de la utilización de algoritmos de decisión heurísticos para el cálculo de los niveles de llenado, como el algoritmo de comparación de umbrales. Así, aunque se sacrificaba encontrar una solución óptima para todos los casos, se ganaba en tiempo de computación y se reducía la complejidad de las operaciones a llevar a cabo. Pero después, como una idea nueva a explorar, se propuso el uso algoritmos de aprendizaje supervisado, concretamente de *SVMs*, que aunque muy costosos computacionalmente, aseguraban encontrar una solución óptima, si existiera, y por tanto así se reducía el error que se iba a cometer.

A continuación se explica brevemente en qué consisten estas *SVMs*, y por qué no se acabaron implementando.

3.1.1 Que es el aprendizaje supervisado

Antes de entrar en qué es el aprendizaje supervisado es útil entender, al menos básicamente, en que consiste el aprendizaje máquina. Este campo, que evolucionó a partir de la teoría del aprendizaje computacional, busca la construcción de algoritmos que permitan aprender y hacer predicciones a partir de unos datos. Estos algoritmos lo que hacen es crear un modelo a partir de los datos de entrada, lo que les permite tomar decisiones y hacer predicciones sobre ese tipo de datos [1]. Y una de las categorías que comprende el aprendizaje máquina es el aprendizaje supervisado.

El aprendizaje supervisado consiste en inferir una función a partir de unos datos etiquetados de entrada, o datos de entrenamiento, los cuales se componen de dos partes: el objeto de entrada (generalmente un vector de dimensión N) y el valor deseado de salida, o señal supervisora. Es esta señal la peculiaridad del aprendizaje supervisado, pues es la que aporta una información extra, se está "supervisando" al algoritmo. Por lo tanto un algoritmo de aprendizaje supervisado analiza estos datos y, tras buscar patrones, genera una función que modela el problema a solucionar.

3.1.2 SVMs

Las SVMs son modelos de aprendizaje supervisado que, en pocas palabras, buscan a partir de unos datos de entrenamiento un hiperplano que permita separarles óptimamente (para el caso de decisión) o se ajuste adecuadamente a estos datos (para el caso de regresión). Para ello hace uso de ciertas muestras, denominadas vectores soporte, las cuales definen el hiperplano que se busca. Es este problema, lo que se trata de obtener es una función $f(x)$ que se ajuste todo lo posible a unos datos de entrenamiento dados y que permita, al introducir nuevas entradas, obtener el nivel de llenado. Esto encaja perfectamente con el caso de regresión antes mencionado, siendo $f(x)$ el hiperplano a encontrar. Para este problema los objetos de entrada, o inputs, serían los valores de potencia de las 15 ventanas, la temperatura, la estima de la varianza de ruido, el tipo de contenedor... y todos los datos que se consideraran útiles. En el caso de los datos de entrenamiento, habría que incluir la "señal supervisora", o etiqueta, que sería la ventana de llenado correspondiente a cada input. Se considera que los objetos de entrada son de dimensión d. A continuación se procede a explicar brevemente cómo funcionan las SVMs para los problemas de regresión. Para más información sobre estos modelos se recomienda leer [2] y [3].

La función a buscar se define como

$$f(\vec{x}) = \langle \vec{v}, \vec{x} \rangle + b, \tag{3.1}$$

con $\vec{v}$ siendo el hiperplano en $\mathbf{R}^d$ formado por los vectores soporte**,** $b \in \mathbf{R}$, y donde $\langle .,. \rangle$ representa el producto interno en $\mathbf{R}^d$.

Un problema es que muchas veces no es posible ajustar los vectores de entrada de manera lineal (en otras palabras, con un hiperplano en $\mathbf{R}^d$). Para esto, las SVMs lo que hacen es elevar el problema a dimensiones superiores, permitiendo encontrar una solución que, aunque en nuestro dominio $\mathbf{R}^d$ no sea lineal, si lo sea en el nuevo dominio. Para ello se hace uso de un método denominado truco kernel (*kernel trick*). Este *kernel trick* consiste en hacer uso de un tipo de funciones denominadas funciones kernel, $K(\vec{v}, \vec{w})$, las cuales son funciones $K: R^N \text{x}\, R^N \rightarrow R$. Realmente lo que hacen es el producto interno entre el vector $\vec{v}$ y el vector $\vec{w}$, pero añadiendo un detalle más, que es lo que las hace realmente útiles: las operaciones en el espacio N tambien se computan automáticamente en el espacio superior M, sin hacer falta transformar a $\vec{v}$ y $\vec{w}$ a dicha dimensión. De esta forma el truco kernel evita los problemas derivados de trabajar en dimensiones superiores [4].

Mediante este truco, las SVMs consiguen encontrar los l vectores soporte $\vec{x}_i$ y sus correspondientes multiplicadores de Lagrange $\left(\overrightarrow{\theta_l}, \overrightarrow{\theta_l^*}\right)$ para $i = 1, \ldots, l$. Una vez obtenidos, el procedimiento es sencillo, ya que la función $f(x)$ se calcula directamente como

$$f(\vec{x}) = \sum_{i=1}^{l} \left(\overrightarrow{\theta_l} - \overrightarrow{\theta_l^*}\right) K(\overrightarrow{x_l}, \vec{x}) - \frac{1}{2} \sum_{i=1}^{l} \left(\overrightarrow{\theta_l} - \overrightarrow{\theta_l^*}\right) \left(K(\overrightarrow{x_l}, \overrightarrow{x_r}) + K(\overrightarrow{x_l}, \overrightarrow{x_s})\right), \tag{3.2}$$

donde

$$\sum_{i=1}^{l} \left(\overrightarrow{\theta_l} - \overrightarrow{\theta_l^*}\right) K(\overrightarrow{x_l}, \vec{x}) = \langle \vec{w}, \vec{x} \rangle,$$

$$-\frac{1}{2} \sum_{i=1}^{l} \left(\overrightarrow{\theta_l} - \overrightarrow{\theta_l^*}\right) \left(K(\overrightarrow{x_l}, \overrightarrow{x_r}) + K(\overrightarrow{x_l}, \overrightarrow{x_s})\right) = b,$$

que si se sustituyen en (3.2) da lugar a la ecuación (3.1) ya conocida. Introduciendo el término b en la función kernel, la función de regresión queda de la siguiente forma

$$f(x) = \sum_{i=1}^{l} (\theta_i - \theta_i^*) K(x_i, x) \tag{3.3}$$

3.1.3 Implementación: Problemas encontrados

Por tanto, si el sensor es capaz de implementar esta función (ec. 3.3), la solución es tan sencilla como aplicar la expresión para cada nuevo vector de entrada. Por tanto, se instaló una *toolbox* de Matlab que permite el uso de SVMs [6]. Tras introducirse brevemente en su manejo, se realizaron unas pruebas sencillas con datos artificiales y vectores de dimensión 18, ejemplificando los 18 elementos que se usarían inicialmente (los 15 valores de potencia de cada ventana, la temperatura, la estima de la varianza de ruido y el tipo de contenedor). Los resultados, aunque en principio consistentes, revelaron un problema. Y es que el número de vectores soporte se encontraba entorno a los 100-200 (dependiendo de los datos), y se ha de tener en cuenta que para implementar esta función es necesario guardar esos 100-200 vectores soporte (cada uno de ellos de dimensión 18) y los 100-200 multiplicadores de Lagrange duales. En un ordenador puede que no haya problemas de memoria, pero nuestro sensor dispone de un registro de memoria limitado. Y, partiendo del hecho de que hubo problemas para poder guardar las 500 muestras, poder guardar tantos valores no se planteaba como una opción posible. Por otro lado, aun suponiendo que se dispusiera de memoria suficiente para todos los datos necesarios, aun hacía falta llevar a cabo todas las operaciones (las cuales incluyen cientos de sumas, cientos de multiplicaciones y calcular la función kernel, la cual no suele ser trivial). Esto, para un sensor de tan limitada capacidad, era imposible, y aunque pudiera llevarlo a cabo (tardando los ciclos que fueran), el consumo de batería sería inmenso, resintiendo gravemente la autonomía del sensor. Por estas razones se vio que, aunque las SVMs podrían ser capaces de obtener una solución óptima, su uso no era posible debido a las limitaciones que imponía el problema. Por tanto se decidió seguir con los algoritmos heurísticos los cuales, aunque se requiera de más esfuerzo y tiempo para su optimización, se adecuan mejor a las características de los sensores con los que se trabaja.

3.2 Algoritmos de decisión heurísticos

Viendo que la primera opción era inviable dado el hardware del que se dispone, se pasa a describir los algoritmos heurísticos que había planteados. Estos algoritmos son el algoritmo de comparación con umbrales, el algoritmo de compensación de atenuación, el algoritmo diferencial y el algoritmo diferencial ponderado.

3.2.1 Algoritmo de comparación con umbrales

Este algoritmo se basa en ir comparando los valores de cada ventana con otros valores fijos, denominados umbrales, de manera consecutiva empezando por la ventana 1. En el momento que uno de los valores supere el del umbral, se tomará la ventana a la que corresponde dicho valor como la correcta, dejando sin comprobar los siguientes valores. Esto permite tomar siempre la ventana que implica que el contenedor está más lleno, lo que en parte nos interesa, ya que en caso de cometerse un error en la estimación del nivel de llenado de los contenedores, resulta preferible en relación al efecto sobre la limpieza vial considerar que el contenedor está lleno cuando realmente está vacío, pues así se evita la acumulación de basura alrededor del contenedor. Como se comentó anteriormente en el capítulo 2, este algoritmo fue el primero en ser usado, ya que es el más común para aplicaciones de detección de objetos mediante ultrasonidos y por lo general viene implementado en el propio sensor, pudiendo programar los umbrales para adaptarlo a la aplicación deseada (como ejemplo, véase [5]).

Una manera de expresar este algoritmo matemáticamente es mediante la expresión

$$D = \min\big(U(k)\big),$$

donde D la decisión tomada y $U(k)$ se define como

$$U(k) = \begin{cases} k & w_k > u_k \\ 15 & en\ otro\ caso \end{cases},$$

siendo w_k el valor de la ventana k y u_k el valor del umbral correspondiente a la ventana k. Lo que se consigue con esta expresión es elegir el mínimo valor de la función $U(k)$, que será el primer k para el que $w_k > u_k$. Estos umbrales u_k se habían obtenido de manera heurística, observando los valores típicos para cada ventana en muchas realizaciones, y deduciendo cuáles de ellos se amoldarían mejor al problema. Esto, sin embargo, conlleva un problema grave: los umbrales son fijos, con lo cual no responde

correctamente a las conocidas variaciones que ocurren en los niveles de la señal, ya sean debidas a la temperatura, la diferencia entre sensores, u otras causas. Por tanto, aunque en general pueda responder muy bien, habrá ocasiones en que cometerá graves errores, con lo que no es la mejor opción. Además, aunque el que sea conservador en cuanto a la ventana de llenado puede ser una ventaja en ocasiones, otras veces produce errores indeseados. Y es que puede acabar eligiendo una ventana porque superaba ligeramente el umbral (debido a ruido u otras causas), e ignorar la correcta la cual se encuentra entre las siguientes, pues detiene el análisis en el momento de elegir una ventana como la correcta.

3.2.2. Algoritmo de compensación de atenuación

Este algoritmo, como su propio nombre indica, busca contrarrestar el efecto de la atenuación que se produce en la onda acústica al propagarse por el espacio. Para ello, lo que se hace es normalizar la atenuación sufrida en el centro de cada ventana respecto a la sufrida en el centro de la ventana 1. Esto se hace principalmente por dos razones. Primeramente, no se disponen de los datos suficientes para calcular los valores de atenuación absolutos (las ganancias varían respecto a cada sensor, por ejemplo). Por otro lado no es necesario conocer dichos valores absolutos, puesto que lo importante es obtener la atenuación relativa para poder compensar los valores de cada ventana de manera adecuada. Estos valores se calculan respecto un punto, que se denominara d_1. Para este caso se considera que $d_1 = 5$ cm, que se corresponde aproximadamente con el punto medio de la primera ventana.

Para calcular los demás puntos se procede a usar la expresión

$$d_k = d_1 + 33kd_0 = d_1 + 11.32, \quad k = 2, \dots, 15$$

donde d_k es la distancia para la ventana k, d_0 es 0.343 cm/muestras (ec. 2.2) y el 33 se corresponde con el número de muestras para cada ventana, como ya se vio en el capítulo 2. A partir de estas distancias se calcula la atenuación para cada ventana mediante la expresión

$$a_k = \left(\frac{d_k}{d_1}\right)^{\rho}, \qquad k = 1, \dots, 15$$

donde a_k es la atenuación normalizada para la ventana k, d_k es la distancia al centro de la ventana k y ρ es el coeficiente de atenuación. Este coeficiente en condiciones ideales, al igual que en las ondas electromagnéticas, es igual a dos. Pero debido a las características propias del contenedor (temperatura, humedad, etc), este parámetro se ve modificado, tomando valores de entre 1.4 a 1.8. Por tanto es este parámetro el que permite ajustar en mayor o menor medida el algoritmo, permitiendo que tome más en consideración las distancias más lejanas u otorgando más fuerza las cercanas.

Una vez obtenido el valor de atenuación adecuado para cada ventana, se procede a compensar los valores de potencia, y una vez realizado esto, se busca el máximo. El algoritmo por tanto quedaría como

$$D = \text{argmax}_k(w_k * a_k) = \underset{k}{\text{argmax}}\left(w_k * \left(\frac{d_k}{d_1}\right)^{\rho}\right). \quad k = 1, \dots, 15$$

En la figura 5 se ve la comparación entre una señal sin modificar, y la misma compensada por los valores de atenuación. Se ve como, aunque en la figura 5.a se podría decidir más o menos sin dudas la ventana de decisión es la 7, en figura 5.b se ve claramente y sin ningún género de duda.

Pero como los demás algoritmos aquí mostrados, también tiene problemas. El más importante de ellos: asume un comportamiento ideal tanto de transmisor como el receptor, ya que únicamente tiene en cuenta el fenómeno de la atenuación de la propagación. Y es que, aunque podría funcionar correctamente e incluso superar a casi cualquier otro algoritmo en condiciones ideales, deja de lado muchos elementos que podrían trastocar completamente su funcionamiento. Un claro ejemplo es el ruido. Cuando se amplifica el valor de la ventana durante la compensación de la atenuación, se está dejando de lado completamente el hecho de que esa ventana es ruidosa, y por tanto se está amplificado tanto el valor real como el valor ruidoso, cometiendo un error bastante grave. Si se da el caso de que en la ventana no haya señal, el error es aún mayor, ya que únicamente se incrementa el ruido. Otro problema que puede ocurrir es causa de los valores de potencia no deseados en la ventana 4, ya que al compensar con a_4, un valor que en muchos casos no influenciaría apenas se ve aumentado considerablemente, pudiendo llegar a elevarse por encima del valor de potencia de la ventana correcta. Esto también puede pasar con el *crosstalk* en la ventana 2, aunque en menor medida ya que a_2 suele ser pequeña respecto a las demás a_k.

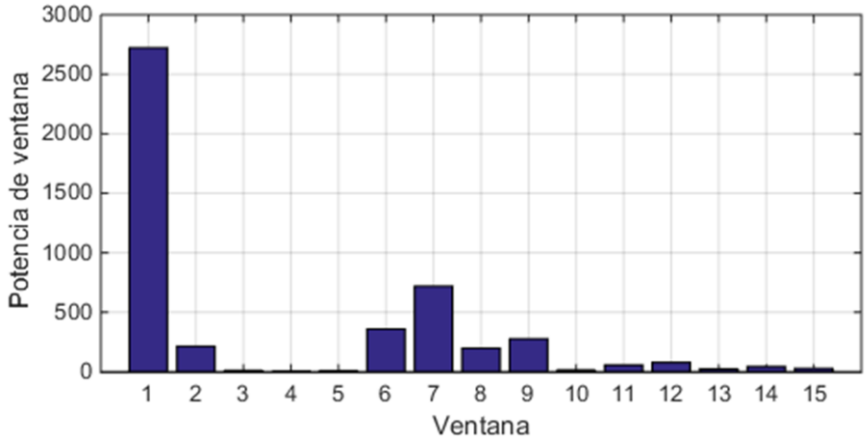

Figura 5.a. Ventanas sin compensar

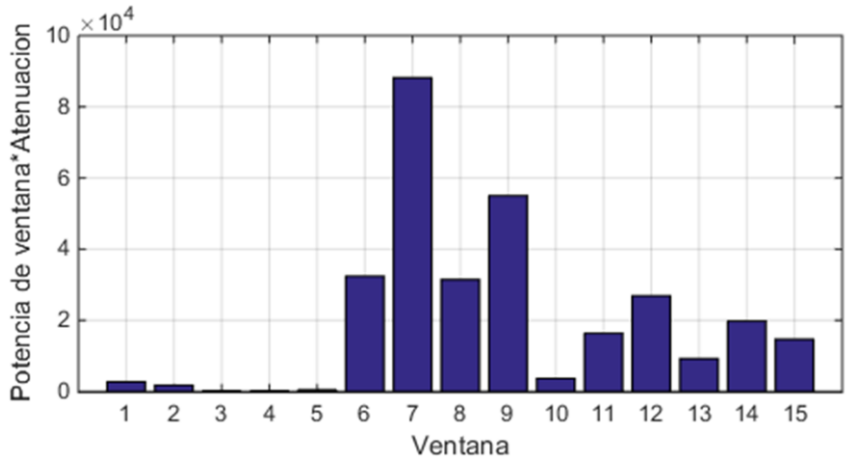

Figura 5.b. Ventanas con atenuación compensada

3.2.3. Algoritmo diferencial

Este algoritmo se basa en calcular la diferencia entre una ventana y la inmediatamente anterior, y a partir de ahí, calcular el máximo de entre todos los valores. Por tanto, busca el mayor salto (positivo) que se da en la captura, y da como válida la ventana en la que se produce dicho salto. La decisión para este algoritmo se expresaría como

$$D = \underset{k}{\operatorname{argmax}}(w_k - w_{k-1}), \qquad k \epsilon\, 2,3,\ldots,15$$

donde w_k son los valores de cada ventana k para una captura dada. En el caso de la ventana 1, para la cual no hay ninguna ventana que la preceda, se usa un umbral fijo con el que se compara. En caso de superarlo se toma la decisión de que la captura corresponde a la ventana 1, independientemente del valor obtenido mediante el

algoritmo diferencial. La principal característica de este algoritmo viene dada por el hecho de que no se ve afectado por varianzas en la ganancia del sistema, pues mientras se mantenga la proporción entre ventanas se podrá discernir en cual hay un salto mayor.

Sin embargo, tiene ciertos problemas derivados de la física del problema y de los sensores. Primeramente, si se da el caso de que el contenedor está vacío, al realizar diferencias entre datos que no son más que ruido se obtendrán resultados muy irregulares y básicamente aleatorios; y por tanto la decisión vendrá en función de que ventana era más alta respecto a su adyacente en el preciso instante que el sensor hizo la medición, cometiendo en la mayoría de los casos un error. Por otro lado, como ya se comentó, en ciertos sensores se había observado que, aun a pesar de estar el contenedor completamente vacío, se encontraba cierta potencia en las ventanas 4 y 5; esto podría provocar que en esos sensores el algoritmo observe ventana 4 o 5 cuando debería ser 15.

3.2.4. Algoritmo diferencial ponderado

Basado en el algoritmo diferencial, este algoritmo busca añadir algún parámetro que permita solucionar los problemas que se producían en el algoritmo diferencial.

Así, se añadió un parámetro más al algoritmo, que restaría a la diferencia de ventanas un valor distinto para cada una de ellas. Este parámetro se denominó gamma, y buscaría solucionar los dos grandes problemas antes mencionados, reduciendo los valores no deseados de la ventana 4 e intentando mejorar las prestaciones del algoritmo en los casos en los que el contenedor se encuentre vacío. Se denotaría por el símbolo $\vec{\gamma}$, y consistiría en 14 valores, uno para cada ventana exceptuando la primera, ya que esta se compara con un umbral.

De esta manera, el algoritmo quedaría como

$$D = \underset{k}{\operatorname{argmax}}(w_k - w_{k-1} - \gamma_k). \qquad k \in 2,3,\dots,15$$

Para este vector $\vec{\gamma}$ se habían ideado unos valores heurísticos de la misma forma que se calcularon los umbrales para el algoritmo de comparación de umbrales.

3.3 Determinación del error de decisión

Uno de los objetivos de este trabajo es fijar un criterio de comparación justo que permita comparar las prestaciones de los diversos algoritmos entre ellos y que emule las prestaciones que se pretenden conseguir. Para ello, se planteó medir el error cometido en la toma de la decisión, eligiendo varias formas de calcular este error y observando cual se podría amoldar mejor al problema. Debido a que no es necesaria demasiada precisión a la hora de decidir el nivel de llenado, equivocarse por una ventana o dos no debería ser un problema demasiado grave, y por lo tanto debería aportar muy poco en el error.

En este apartado se hablará siempre de errores absolutos, pues lo que se busca es una forma de comparar los algoritmos entre sí. Para simplificar y no reiterarlo cada vez que se comenten las expresiones de los errores, se utilizará V para referirse al valor de llenado real, y D para hablar de la decisión del algoritmo de decisión. Para hablar del error se denotara simplemente como E, añadiendo un subíndice para diferenciar que tipo de error es.

Los errores que se plantearon son tres: el error cuadrático, el error lineal, y el error ε-*insensitive*. A continuación se detalla cada uno más detenidamente.

3.3.1 Error cuadrático

Uno de los errores más utilizados en tratamiento de señal, en gran parte debido a que su expresión es derivable, y por tanto, permite la búsqueda de mínimos que permiten minimizar el error. Se expresa de la mediante la expresión

$$E_c = (V - D)^2.$$

Sin embargo, para adaptarse a este problema no es muy útil, pues como se puede apreciar en la figura 6 el crecimiento de error a medida que hay más diferencias entre ventanas es muy alto; esto da excesiva importancia a las equivocaciones por muchas ventanas, y deja a algo anecdótico las equivocaciones por pocas.

Y es que un fallo por bastantes ventanas tiene que tenerse muy en cuenta, y es un error grave, pero también es importante que las equivocaciones por pocas ventanas aporten de manera significativa al error total.

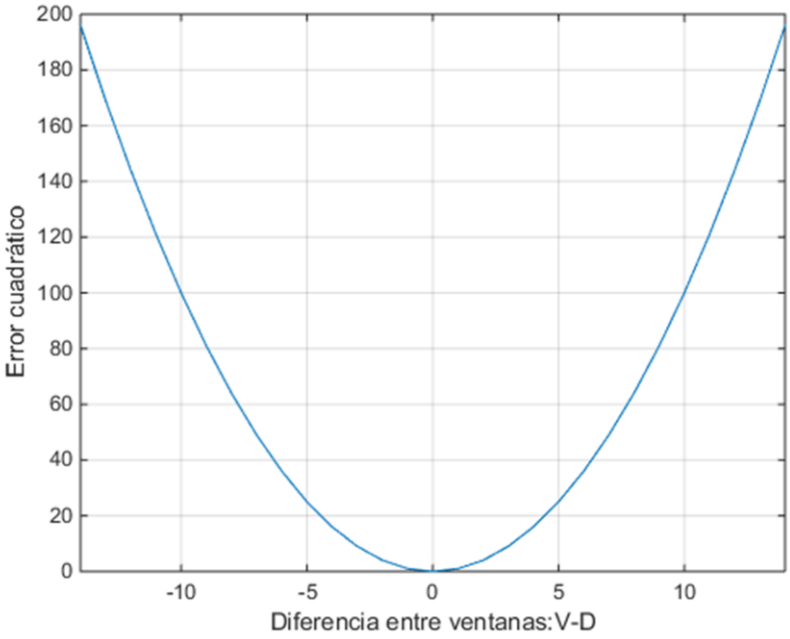

Figura 6. Error cuadrático

3.3.2. Error lineal

Este error se ajusta mejor a la heurística del problema, dando un valor alto a un error por muchas ventanas, pero sin que reste importancia a los fallos por pocas ventanas, como ocurría en el error cuadrático. En la figura 7 se aprecia perfectamente como todos los valores que puede tomar este error son del mismo orden. Otra ventaja añadida es que si se obtiene el error para una gran cantidad de realizaciones, y se normaliza en base al número de estas, el resultado obtenido es el número medio de ventanas por las que se equivoca el algoritmo, lo cual facilita el análisis a simple vista.

Su expresión es

$$E_L = |V - D|.$$

Sin embargo, en el problema a resolver equivocarse por una ventana, e incluso por dos, no debería ser algo demasiado grave, pues lo que se busca es saber cómo de lleno está el cubo, sin necesidad de ser demasiado preciso. Por tanto se plantea la siguiente opción.

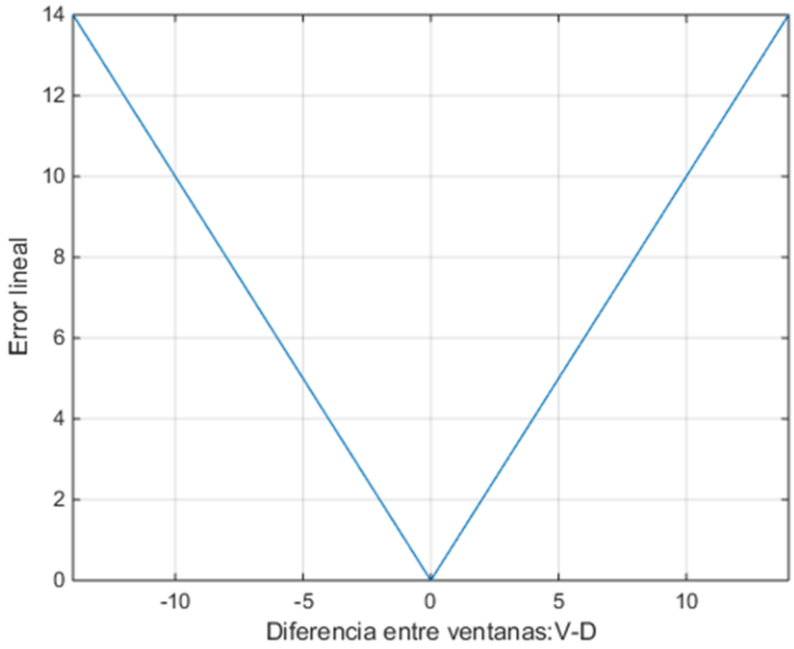

Figura 7. Error lineal

3.3.3. Error *ε-insensitive*

Este error, cuya idea se obtuvo de [3], se acerca mucho más a la idea buscada. Esto es debido a que además de mantener la idea de que un error entre varias ventanas influye más que uno por pocas, y que independientemente de cómo de grande haya sido la equivocación esta es siempre de un orden similar, este error añade una nueva característica.

Y es que usando este error, una equivocación por ε ventanas no se tiene en cuenta, a todos los efectos se considera un acierto. En el caso a tratar, se ha elegido ε=1 (dando lugar a una función como la que se aprecia en la figura 8) ya que eso da un margen de unos ±11cm, lo cual es un error aceptable. Se podría plantear ε=2, pero podría dar lugar

a problemas cuando el contenedor se encuentra muy lleno, y ya se estaría trabajando con un margen de error muy amplio.

La expresión del error seria

$$E_\varepsilon = \max(0, |V - D| - \varepsilon).$$

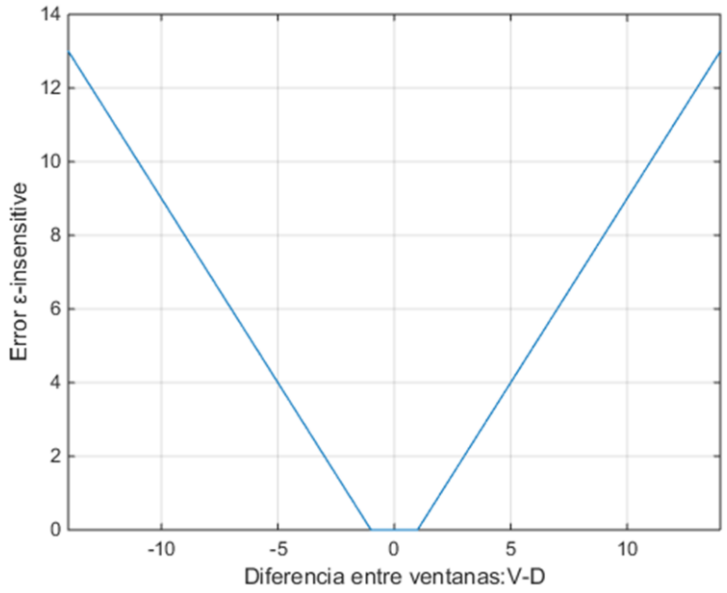

Figura 8. Error ε-insensitive con ε=1

Debido a sus cualidades, este fue el error que se eligió para comparar los diferentes algoritmos entre sí. Era el que mejor se amoldaba al problema, ya que ofrece un margen de error (recordamos que no se busca una gran precisión) y proporciona unos valores equiparables entre si independientemente de la equivocación cometida en la decisión. El ε que se decidió utilizar, como ya se ha comentado previamente, fue ε=1, ya que permite dar un margen tolerable sin que llegue a ser demasiado amplio.

3.4 Registros etiquetados

Una vez explicados los diversos algoritmos que se analizarán y los errores que se utilizarán para comprobar la eficiencia de estos algoritmos, aparece la problemática de necesitar una base de datos etiquetados con los que poder hacer pruebas. En este caso en

particular, es necesaria una gran base de datos debido a que hace falta que cubra todos los casos de llenado, y a la vez, que tenga medidas de suficientes sensores distintos.

En otros casos, obtener una base de datos etiquetada y fiable es una tarea sencilla, que no requiere de demasiado esfuerzo. Para este problema, sin embargo, requeriría hacer pruebas con un contenedor con varios niveles de llenado y varios sensores. Y aun así sería insuficiente, pues como ya se ha comentado anteriormente, las medidas de los sensores difieren bastante entre sí, por electrónica, temperatura u otras razones. Esto es igual para los llenados de los contenedores pues pueden variar completamente ya no solo porque sean de papel o envases, sino por los propios residuos, que son tremendamente irregulares. Esto genera, como ya se ha comentado anteriormente, que la superficie de contacto con la onda sonara emitida por el sensor sea altamente variable. Con lo cual no se podría cubrir todo el espectro de posibilidades.

Otra opción sería ir comprobando contenedores uno a uno a la hora que el sensor toma la medida, midiendo su nivel de llenado, pero esto sería tremendamente costoso en esfuerzo aparte de muy ineficiente. Además de que si se busca una base de datos suficientemente amplia y variada, llevaría un tiempo desproporcionadamente alto.

Por lo tanto, se llega a la conclusión de que la mejor manera, aunque no la ideal, es hacer un etiquetado indirecto de los datos disponibles. Este etiquetado se plantea inicialmente hacerlo a “ojo”, observando para cada realización los 15 valores de potencia de cada ventana (que a partir de ahora denominaremos registro) y decidiendo en función de estos niveles.

Sin embargo, esto está sujeto a varios problemas, siendo el primero lo subjetivo que puede llegar a ser la decisión en algunos registros. Aunque la mayoría de ellos son claros y prácticamente cualquier persona con conocimiento del problema llegaría a la misma conclusión, otros muchos podrían llegar a ser objeto de debate. Un claro ejemplo son aquellos registros en los que aparecen dos o más picos, usualmente debidos a obstáculos inclinados. En estos casos la decisión es ambigua, pues se puede tanto optar por una decisión conservadora, eligiendo la ventana que indica que el contenedor está más lleno, como por una decisión que favorezca el máximo entre los susodichos picos, independientemente de la posición. Otra opción a medio camino es buscar la media

entre la ventana más alta y la más baja, con los problemas que puede acarrear esto, ya que el valor no sería estrictamente real.

A pesar de todo esto, se llevó a cabo el etiquetado de una cantidad de registros que pudiera considerarse aceptable, para así poder hacer unas pruebas iniciales que permitieran observar el comportamiento de los algoritmos. De esta manera, se etiquetaron 1025 registros de 22 sensores distintos y que contenían realizaciones para las 15 ventanas, aunque no en la misma proporción. Durante el etiquetado se tomó un criterio conservador, eligiendo, en el caso de que hubiera dos picos de potencia o más, la ventana que implicaba que el contenedor estaba más lleno.

Una vez obtenida esta base de datos, se pudo comenzar el análisis de los algoritmos explicados anteriormente, lo que permitiría elegir el algoritmo más adecuado para el problema planteado.

3.5 Búsqueda de algoritmo adecuado

Una vez explicados adecuadamente los algoritmos que había sobre la mesa, la métrica que se pretendía utilizar a la hora de compararlos y la problemática de obtener una base de datos adecuada, se procede a explicar el análisis que se llevó a cabo, con la intención de buscar que algoritmo funcionaría mejor para este problema.

3.5.1 Análisis temporal. Predicciones de comportamiento

Para ello, se realizó un análisis inicial, intentando llegar a algunas conclusiones previas antes de analizar el error mediante la base de datos. Por tanto, se recapituló todo lo que se conocía de cada algoritmo, y se hizo un estudio temporal para cada uno de ellos. Este estudio temporal consistía en analizar los registros que había transmitido un sensor durante un determinado espacio de tiempo, y utilizar los cuatro algoritmos a estudiar con dichos registros. De esta manera se pudo ver cómo sería la evolución del llenado del cubo en función de cada algoritmo. Si el resultado seguía una sucesión lógica, en la que se apreciara como se iba llenando el cubo, podría ser viable. Sin embargo, si los resultados eran demasiado erráticos, no debería ser tenido en cuenta. En la figura 9 se muestra un ejemplo de este análisis para un cubo concreto, y para los cuatro algoritmos.

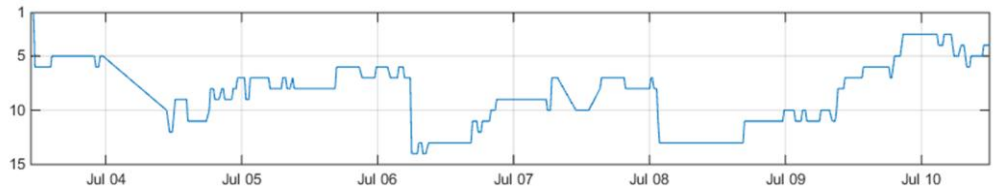

Fig 9.a. Evolución del llenado del cubo con el algoritmo de comparación de umbrales. El eje X representa el tiempo en días y el eje Y la ventana de llenado

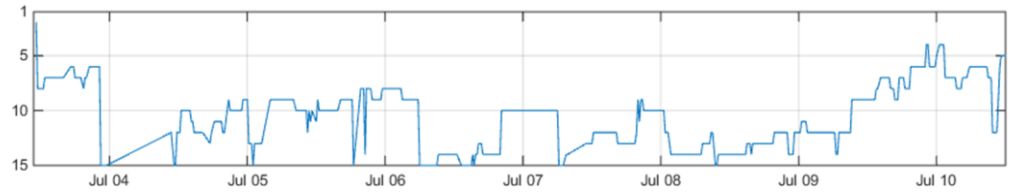

Fig 9.b. Evolución del llenado del cubo con el algoritmo de compensación de atenuación

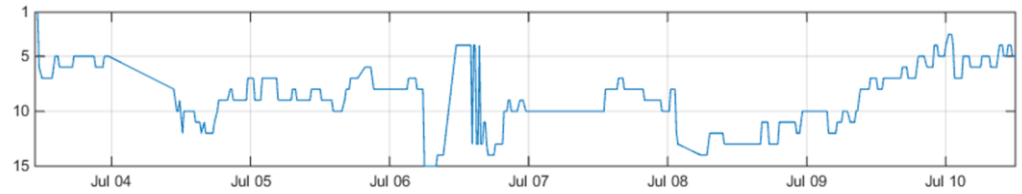

Fig 9.c. Evolución del llenado del cubo con el algoritmo diferencial

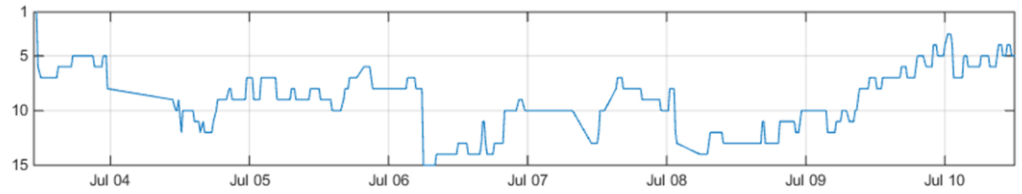

Fig 9.d. Evolución del llenado del cubo con el algoritmo diferencial ponderado

A simple vista todos parecen seguir un orden lógico, viendo como el cubo se va llenando poco a poco y se vacía hasta en tres ocasiones, con lo que no se podría desechar ninguno en primera instancia. A partir de estos resultados, y del conocimiento previo que se tenía de cada algoritmo se pudieron plantear las siguientes predicciones.

Sobre el algoritmo de comparación de umbrales, al haber estado en funcionamiento durante un tiempo se conocen sus prestaciones. Este algoritmo comete muchos errores, debidos sobre todo a su incapacidad para adaptarse a las varianzas de la ganancia del sistema. Por tanto, y aunque el análisis temporal pueda parecer adecuado, lo cierto es

que este algoritmo no es viable, y los resultados que obtengan solo se utilizarán como una referencia a superar.

En cuanto al decisor de compensación de atenuación, aunque permite ajustarse de manera dinámica a las variaciones de la ganancia, el problema que tiene es bastante grave. No tener en cuenta elementos tan importantes como el ruido, a pesar de poder intentar ajustar el parámetro ρ para mejorar su comportamiento, hacen que no se espere un gran rendimiento por parte de este algoritmo. A pesar de esto si se espera que mejore los resultados del decisor de comparación de umbrales, pues al contrario que este último, sí que es dinámico y se amolda a la situación del sensor (niveles de energía) en cada momento. En el análisis temporal se percibió una tendencia bastante clara a elegir ventanas altas como nivel de llenado. Esto puede acabar siendo un problema bastante grave.

Respecto al algoritmo diferencial, al igual que el de compensación de atenuación su capacidad de no verse influido por las variaciones en la energía de la señal es un punto a favor. Sin embargo sus problemas, los cuales no se pueden evitar de ninguna forma, también son graves, con lo que se espera que mejore los resultados del algoritmo de comparación de umbrales, pero no que vaya a tener un gran comportamiento. En el análisis temporal se observó un funcionamiento adecuado, aunque en ciertos momentos se veían rápidos cambios en la decisión, usualmente involucrando a las ventanas 4 y 5 en ellos, viéndose que el fenómeno de valores de potencia en estas ventanas tenía mucho que ver.

Y es en el análisis del algoritmo diferencial ponderado donde se vio un mejor comportamiento, reduciéndose esas variaciones en gran medida, con lo que se puede suponer a simple vista que los $\vec{\gamma}$ si corrigen este error. A parte de esto, del algoritmo diferencial ponderado se espera un mejor comportamiento en general, ya que al fin y al cabo es igual que el algoritmo diferencial, solo que tratando de compensar las carencias de este último. Por tanto, si los valores de $\vec{\gamma}$ que se obtuvieron heurísticamente son adecuados (y el análisis temporal apunta a que sí), es muy posible que mejore los resultados del algoritmo diferencial y del algoritmo de comparación de umbrales.

3.5.2 Calculo de error. Análisis de resultados

Una vez planteado que se podía esperar de cada sensor, se procedió a analizar la base de datos etiquetada con cada uno de ellos, calculándose el error medio por registro que cometía cada algoritmo mediante el uso del error ε-insensitive. A continuación se muestran los resultados para la base de datos de 1025 registros.

Algoritmo	Error medio
Algoritmo de comparación de umbrales	1.088
Algoritmo de compensación de atenuación	1.737
Algoritmo diferencial	0.5366
Algoritmo diferencial ponderado	0.5024

Tabla 1. Error medio para cada algoritmo

Estos resultados en algunos casos difieren respecto a lo que se había previsto, concretamente en cuanto al algoritmo de compensación de atenuación. Sin embargo, una vez que se analizaron los registros y se estudió el porqué de los errores de cada algoritmo se vio mucho más claro. Seguidamente se analizan los resultados y se explica el porqué de los mismos.

Primeramente, el error del algoritmo de comparación de umbrales da un error que se podría calificar como no demasiado malo, pero como ya se conocía su mal funcionamiento y anteriormente se comentó que se iba a utilizar como referencia, no se le dio demasiada importancia.

Es el error del algoritmo de compensación de atenuación el que difiere sobremanera de lo que se esperaba de él, pues se equivoca un 70% más que el algoritmo de comparación de umbrales, el cual no tiene grandes prestaciones como ya se sabe. Fue tras observar algunas realizaciones cuando se entendió porque. Y es que, como se había vislumbrado durante el análisis temporal, el algoritmo de compensación de atenuación da demasiada fuerza a las ventanas más bajas, haciendo parecer que un ruido ligeramente alto es una señal. Esto ocurre sobre todo en los casos en los que la potencia de la ventana que debería ser decidida no es muy alta, como es el caso en la figura 10, donde la ventana a decidir sería la 7 (figura 10.a), y tras la compensación la que se decide es la 14 (figura 10.b).

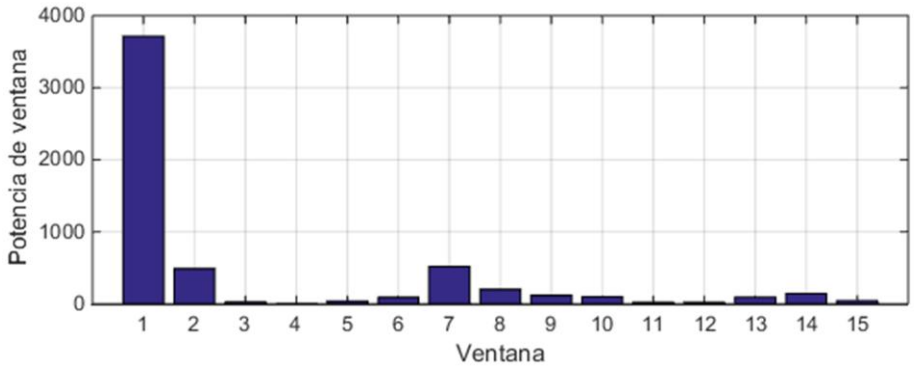

Figura 10.a. Ventanas sin compensar

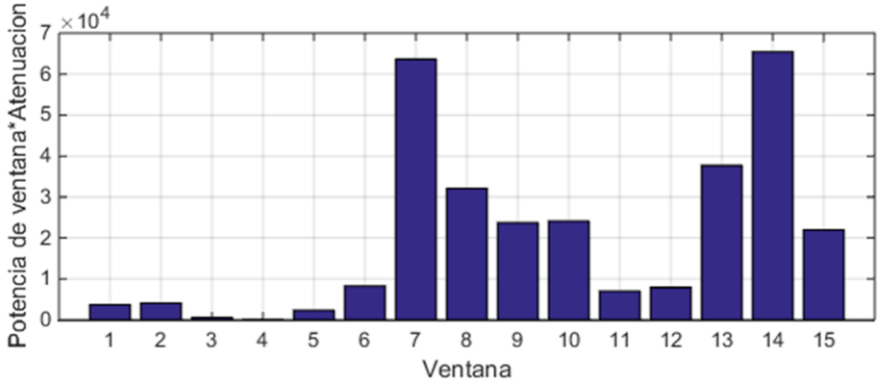

Figura 10.b. Ventanas con atenuación compensada

En lo que se refiere al algoritmo diferencial, se aprecia que tiene un buen comportamiento, cometiendo de media la mitad de errores que el algoritmo de comparación de umbrales, lo cual es una gran mejora. Con respecto al algoritmo diferencial ponderado, se ve una leve mejora, de entorno al 7%, respecto al algoritmo diferencial, lo cual indica que el parámetro $\vec{\gamma}$ es útil y funciona correctamente, aunque

no ha mejorado el comportamiento de los errores todo lo que se pretendía. Esta mejora se consiguió sobretodo en cuanto al problema de los valores de potencia en las ventanas 4 y 5, reduciéndolo y evitando estos errores en la mayoría de los casos. Sin embargo, no se vio que arreglara los errores cometidos para registros donde el contenedor se encontraba completamente vacío, aunque en algunos casos aislados sí que se evitaban.

De todas formas, se ha de tener en cuenta que estos errores están basados en una base de datos que ha sido etiquetada indirectamente (lo que implica que posiblemente se hayan cometido errores en la etiquetación debido a la subjetividad del proceso) y con un criterio conservador (el cual también es subjetivo y no tiene por qué ser el ideal). Por tanto, aunque estas cifras sean útiles para ver a grandes rasgos cómo se comportan los algoritmos, no se pueden sacar conclusiones absolutas sobre el comportamiento de cada uno. Si se puede, sin embargo, deducir cual presenta mejores prestaciones y cuál debería ser estudiado más a fondo y optimizado. Y en base a los resultados de la tabla 1, claramente este algoritmo es algoritmo diferencial ponderado. Por tanto, el siguiente paso es optimizarle, buscando unos valores de $\vec{\gamma}$ que reduzcan el error cometido a la vez mejoran los valores de error obtenidos con los valores de $\vec{\gamma}$ heurísticos. En el siguiente capítulo se explicará cómo se utilizó la optimización convexa para intentar lograr esto.

4. Optimización del algoritmo diferencial ponderado

Una vez se determinó el algoritmo diferencial ponderado como más eficaz, al menos potencialmente, se decidió buscar de una manera teórica un parámetro $\vec{\gamma}$ que pudiera cubrir un espectro más amplio de registros, y que pudiera igualar y mejorar las prestaciones que ofrece los valores de $\vec{\gamma}$ heurísticos.

Para ello, se dispuso utilizar la optimización convexa, principalmente debido a la cantidad de información y recursos que hay sobre este método. De esta manera, a partir de una base de datos etiquetados se buscarían los valores de $\vec{\gamma}$ óptimos, que, si la base de datos resulta ser suficientemente amplia, deberían servir para cualquier caso y/o sensor. Para comprender mejor esto, se procederá a explicar brevemente en que consiste la optimización convexa y en qué casos se puede utilizar.

4.1 Optimización Convexa

Siguiendo a Boyd y Vandenberghe [7], un problema de optimización convexa tiene la siguiente forma

$$\begin{aligned} &\text{minimizar} && f_0(x), \\ &\text{sujeto a} && f_i(x) \leq b_i, \quad i = 1, \dots, m \end{aligned} \tag{4.1}$$

donde el vector $x = (x_1, \dots x_n)$ es la variable de optimización del problema, $f_0: \mathbf{R}^n \to \mathbf{R}$ es la función objetivo, las funciones $f_i: \mathbf{R}^n \to \mathbf{R}$, $i = 1, \dots, m$, son las funciones de restricción y las constantes $b_1, \dots, b_m$, son los límites de las restricciones. Un vector x^* se denomina óptimo, o solución del problema (4.1), si posee el menor valor objetivo de entre todos los vectores que satisfacen las restricciones. En otras palabras: para cualquier z con $f_1(z) \leq b_1, \dots, f_m(z) \leq b_m$, se tiene que $f_0(z) \geq f(x^*)$. Además, se debe cumplir que las funciones objetivo y las de restricciones sean convexas, es decir, que cumplan la inecuación

$$f_i(\alpha x + \beta y) \leq \alpha f_i(x) + \beta f_i(y), \tag{4.2}$$

para todo $x, y \in \mathbf{R}^n$ y todo $\alpha, \beta \in \mathbf{R}$ con $\alpha + \beta = 1, \alpha \geq 0, \beta \geq 0$.

Por tanto, si nuestro problema cumple estas condiciones, se podrá utilizar la optimización convexa, permitiendo por tanto el uso de las múltiples herramientas de las que se dispone para trabajar con estos problemas. Para recordar, nuestro problema es el siguiente:

Se tiene un algoritmo de decisión que da los valores de ventana de llenado $D[n]$ según la expresión

$$D[n] = \underset{k}{\text{argmax}}(w_k[n] - w_{k-1}[n] - \gamma_k), \qquad k \in 2,3, \dots, 15$$

donde $w_k[n]$ es el valor de la ventana *k* para la realización *n*, y γ_k es el valor de $\vec{\gamma}$ para la ventana k.

Se busca una $\vec{\gamma}$ que minimice la función de error $f(D[n], V[n])$, dada por la expresión

$$E_\varepsilon[n] = f(D[n], V[n]) = \max(0, |D[n] - V[n]| - \varepsilon)$$

A simple vista, formular este problema como un problema convexo no es sencillo. La función objetivo a minimizar debería ser el error, pero ya aquí hay un problema, pues $f(D[n], V[n])$ no es convexa ya que no cumple la condición esencial (4.2). Por tanto, es necesario buscar un problema equivalente que sea convexo, de manera que, si se obtuviera una solución para este problema equivalente, esta fuera también solución del problema original no convexo. A continuación se describe el procedimiento seguido.

4.2 Convexización del problema

Lo que busca el algoritmo diferencial es buscar el máximo entre varios valores, por tanto se puede decir que para una realización n donde $v = V[n]$ es la ventana correcta, se debe cumplir la siguiente inecuación

$$w_v[n] - w_{v-1}[n] - \gamma_v \geq w_k[n] - w_{k-1}[n] - \gamma_k, \qquad \begin{cases} v = V[n] \\ n = 1,2, \dots, N \\ k = 2,3 \dots . 15 \\ v \neq k \end{cases}$$

O, para simplificar, definiendo $x_i[n] = w_i[n] - w_{i-1}[n]$ se obtiene

$$x_v[n] - \gamma_v \geq x_k[n] - \gamma_k. \qquad \begin{cases} v = V[n] \\ n = 1,2,\dots,N \\ k = 2,3 \dots .15 \\ v \neq k \end{cases}$$

Es decir, que $x_v[n] - \gamma_v$ debe ser mayor que su homologo para todas las ventanas que no sean v, es decir, que no sean la correcta, y esto tiene que ocurrir para cada realización. De esta manera se elimina el operando argmax_k de la expresión del algoritmo, creando a cambio 13 condiciones por cada realización *n*.

Pero para poder crear un problema convexo equivalente al original es necesaria una función de coste a minimizar. Así, se crea un vector auxiliar $\vec{\alpha}$ cuyos elementos sean de la forma α_{vk} para todas las combinaciones de v y k, sin repetición, siendo v, k=2, … ,15. Una buena forma de verlo es considerando una matriz Λ de la forma

$$\Lambda = \begin{pmatrix} \alpha_{22} & \cdots & \alpha_{2,15} \\ \vdots & \ddots & \vdots \\ \alpha_{15,2} & \cdots & \alpha_{15,15} \end{pmatrix}.$$

Debido a que no es necesario comparar una ventana consigo misma, los valores de α_{vk} con $v = k$ son innecesarios, con lo que se eliminaría la diagonal de la matriz y se vectorizaría, obteniéndose así el vector $\vec{\alpha}$ con el que se trabajará. Este nuevo vector auxiliar se introduce en las restricciones como

$$x_v[n] - \gamma_v + \alpha_{vk} \geq x_k[n] - \gamma_k. \qquad \begin{cases} v = V[n] \\ n = 1,2,\dots,N \\ k = 2,3 \dots .15 \\ v \neq k \end{cases} \tag{4.3}$$

Idealmente, los valores α_{vk} deberían de ser 0, pues debería ser suficiente con $\vec{\gamma}$. Por tanto, se ha de crear una función de coste, que será la función objetivo, que minimice los valores de $\vec{\alpha}$, por ejemplo

$$f(\alpha) = \sum_{\substack{v,k=2,\dots,15 \\ v \neq k}} |\alpha_{vk}|.$$

Por tanto, estos valores se utilizan para que, durante la optimización del problema, se busquen los valores idóneos de $\vec{\gamma}$, buscando que los α_{vk} sean 0, ya que idealmente es el valor que deberían tener. Sin embargo en ocasiones no se podrán encontrar unos γ_k que cumplan todas las condiciones con $\vec{\alpha} = \vec{0}$. En estos casos algunos α_{vk} (aquellos que correspondan con la restricción que no se pueda satisfacer) tomarán valores distintos de 0 (del orden de los valores típicos de la ventana o más altos), lo que permite ver si una o más restricciones no se han podido cumplir.

Pero a pesar de todo, esto no ajusta exactamente al problema, ya que no se está teniendo en cuenta el error *ε-insensitive* que se consideró apropiado en el capítulo anterior, y que por supuesto se buscaría minimizar. Una manera sencilla de introducir algo similar a este error, es mediante el uso de unos pesos que multiplicarían a $\vec{\alpha}$ en la función de coste, que se llamarán C_{vk}. Estos valores seguirán la expresión

$$C_{vk} = |v - k| - \varepsilon. \qquad v, k \in 2,3, \dots 15 \quad v \neq k$$

Por lo tanto el rango de C para ε=1 sería $C \epsilon [0,12]$, y la expresión de la función objetivo quedaría

$$f(\alpha) = \sum_{\substack{v,k=2,\dots,15 \\ v \neq k}} C_{vk} |\alpha_{vk}|.$$

(4.4)

Lo que se consigue mediante estos pesos es dar más importancia a la minimización de unas alfas frente a otras. Así, aunque se deban minimizar todas las alfas, no se minimizan de igual forma, si no que las que corresponden a alfas con unas v y k mas separadas (o lo que es lo mismo, que las ventanas que se están comparando se encuentran más separadas) se busquen minimizar más. Esto tiene sentido, ya que si por ejemplo asumimos que la ventana 2 es la correcta para la realización n, es más importante que se cumpla que $x_2[n] - \gamma_2$ sea mayor que $x_{15}[n] - \gamma_{15}$ antes que $x_2[n] - \gamma_2$ sea mayor que $x_3[n] - \gamma_3$, puesto que si la segunda condición no se llegara a cumplir no se cometería tanto error. Por tanto se obliga a que las condiciones con v y k separadas sean más importantes de cumplir que las de v y k más cercanas. El hecho de que C pueda valer 0 también permite ignorar en cierta medida las condiciones en los

que la diferencia entre ventanas a comparar sea ε, puesto que el valor de esos α_{vk} no se tendrá en cuenta a la hora de minimizar.

Por tanto, el problema reformulado quedaría de la forma

$$\text{minimizar} \qquad f(\alpha) = \sum_{\substack{v,k=2,\dots,15 \\ v \neq k}} C_{vk} |\alpha_{vk}| \,,$$

$$\text{sujeto a} \qquad x_v[n] - \gamma_v + \alpha_{vk}{}^2 \geq x_k[n] - \gamma_k \,. \qquad \begin{cases} v = V[n] \\ n = 1,2,\dots,N \\ k = 2,3 \dots .15 \\ v \neq k \end{cases}$$

(4.5)

Se puede demostrar fácilmente que tanto $f(\alpha)$ como $\gamma_v - \gamma_k - \alpha_{vk}$ (que serían las f_i que se ven en (4.1), solo que ahora dependen de v y k) cumplen la expresión (4.2) y que por lo tanto son funciones convexas. Por lo tanto se concluye que se puede utilizar la optimización convexa para resolver el problema.

Una vez creado el problema equivalente, solo quedaría elegir la herramienta que permitiría llevar a cabo el cálculo de $\vec{\gamma}$. Para esto hay múltiples posibilidades, pero la que se decidió utilizar en este caso fue la herramienta CVX. Se pretendía introducir una base de datos etiquetada suficientemente amplia tanto en tamaño como en variedad y que mediante ella calculara el vector $\vec{\gamma}$ óptimo que ajusten lo mejor posible las decisiones. Pero antes se explicará brevemente que es CVX, y como funciona.

4.3 CVX

CVX es un conjunto de funciones de modelado para optimización convexa basado en Matlab, permitiendo especificar restricciones y funciones objetivo utilizando la sintaxis típica de Matlab. Por defecto, CVX soporta una aproximación a la optimización convexa llamada programación convexa disciplinada. Esta aproximación busca facilitar el acceso a la programación convexa, reduciendo la necesidad de tener un profundo entendimiento sobre análisis convexo y algoritmos numéricos, el cual es necesario

[2] Para asemejarse más a la expresión vista en (4.1) se han reordenado las variables de manera que queden a un lado todas las variables, y por otro los términos independientes.

normalmente. Así, permite acceder a este tipo de herramientas matemáticas sin tener que introducirse de manera profunda en un tema tan complejo como la optimización convexa. Para más información se puede consultar [8] y [9].

Como buen sistema para la resolución de funciones convexas, a CVX hay que proporcionarle tres parámetros primarios (aunque profundizando se le puedan añadir más elementos para modelar mejor el problema que se quiera solucionar).

- Variables a calcular: Primeramente, a CVX se le deben dar las variables que tendrá que optimizar, indicando su tamaño. En este problema las variables son $\vec{\gamma}$, compuesta por 14 valores, y $\vec{\alpha}$, compuesta por 182 valores $(\alpha_{23}, \alpha_{24}, \dots, \alpha_{32}, \alpha_{34}, \dots, \alpha_{15,13}, \alpha_{15,14})$.

- Función objetivo: Como ya se ha visto antes los problemas de optimización siempre requieren de una función objetivo a minimizar. En este caso la función es la ecuación (4.4)

- Restricciones: Por último, se deben añadir las funciones de restricción. En el caso de este problema, las restricciones son las desigualdades que se han comentado antes en la ecuación (4.3)

  Además, se debe imponer un límite para los valores de $\vec{\gamma}$, que en este caso viene dado por el hecho de que los registros del sensor son de 16 bits, con lo que

  $$0 \leq \gamma_k \leq 65535. \qquad k = 2, \dots, 15$$

  En cuanto al $\vec{\alpha}$, debido a que es una variable auxiliar que solo se utiliza para el cálculo del $\vec{\gamma}$ optimo y no para el cálculo de la decisión, no es necesario imponerle ninguna restricción.

4.4 Implementación del problema equivalente

Tras tener el problema equivalente y comprender el funcionamiento de CVX, se procedió a su implementación.

Primero se realizó un problema más sencillo sin incluir el error, es decir, sin incluir las C_{kj}. De esta manera se pudo depurar los problemas a medida que iban surgiendo de manera más fácil, y así luego el programa se podría extender incluyendo los valores C_{kj}. Por tanto el problema a optimizar era

$$\begin{aligned} &\text{minimizar} && f(\alpha) = \sum |\alpha_{kj}|, && \forall\, k, j \epsilon\, 2{,}3 \ldots..15 && k \neq j \\ &\text{sujeto a} && \gamma_j - \gamma_k + \alpha_{kj} \geq x_j[n] - x_k[n], && \forall\, k, j \epsilon\, 2{,}3 \ldots..15 && k \neq j \\ & && 0 \leq \vec{\gamma} \leq 65535. \end{aligned}$$

Se puede ver que se realizó un cambio de signo en las restricciones por facilidades a la hora de programar, pero no cambia nada a la hora de obtener los resultados. Una vez se acabó de implementar esta simplificación, y tras ver que funcionaba correctamente, se procedió a programar el problema completo. Sin embargo, y aunque a simple vista pueda parecer que el problema se puede escribir en CVX de manera directa, lo cierto es que hubo que hacer ciertas modificaciones. El problema final que se implementó fue el que se observa en la ecuación (4.5), con el cambio de signo antes mencionando

En esta implementación final, $\vec{\alpha}$ ya no es un vector de 182 valores, sino de 156. Y es que para poder emular de manera correcta el error ε-insensitive se debe realizar algo más. A pesar de que cuando $C_{vk} = 0$ se ignoraba el valor del correspondiente α_{vk}, en realidad no se estaba dejando de lado del todo la restricción, sino que se dejaba vía libre para que el algoritmo pusiera los valores que quisiera a esos α_{vk}. Esto provocaba un problema por el cual CVX no convergía adecuadamente, ya que aunque minimizaba la función objetivo, se asignaban unos valores muy altos a esos α_{vk}, los cuales impedían el correcto funcionamiento del programa. Por eso, a la hora de la implementación, estos α_{vk} para $|v - k| = \varepsilon$, así como las restricciones en las que aparecían estos términos, se eliminaron, simplificando de esta manera el algoritmo y facilitando los cálculos. Esta es la razón, por tanto, de que el vector $\vec{\alpha}$ tenga 26 valores menos. Por la misma razón de cada realización se obtienen 11 restricciones, en vez de las 13 que salían anteriormente(a excepción de para las realizaciones de la ventana 2 y la 15, donde se obtienen 12, ya que para estos casos solo se eliminan α_{23} y $\alpha_{15,14}$ respectivamente).

4.5 Resultados

Una vez implementado el problema equivalente en CVX, solo quedaba comprobar que, efectivamente, CVX era capaz de obtener unos valores de $\vec{\gamma}$ que se adecuaran al problema y que minimizaran el error cometido. Por tanto, se procedió a probar el problema equivalente implementado en CVX, al cual a partir de ahora denominaremos método, con la base de datos etiquetada de la que se disponía. Así, se introdujeron los 1025 registros en el programa, obteniéndose tras los cálculos una $\vec{\gamma}$ que, en teoría, debería ser óptima para esta base de datos. Sin embargo, en el proceso también se obtuvieron bastantes α_{vk} no nulos, y con valores bastante altos, lo cual no debería ocurrir. Esto no indicaba que se fueran a obtener buenos resultados, pero aun así se introdujo el vector $\vec{\gamma}$ en el algoritmo diferencial ponderado y se calculó el error medio. El resultado distaba mucho de ser bueno: el error medio era de 1.04, apenas un poco mejor que el error medio que se obtenía con el algoritmo de comparación de umbrales, y prácticamente el doble que se conseguía con el $\vec{\gamma}$ heurístico. La cuestión ahora estaba en donde se encontraba el error, si en el método o en la propia base de datos, pues como ya se ha comentado, al haber sido etiquetada de manera indirecta era posible que se hubieran cometido errores. Por tanto se decidió hacer una prueba para comprobar si realmente el método estaba funcionando correctamente

Para ello, se etiquetó de nuevo la base de datos utilizando el algoritmo diferencial ponderado (ya que era el que se estaba estudiando) con el vector $\vec{\gamma}$ heurístico (que era el que mejor resultados había dado). De esta manera se conseguía que, al haber elegido la ventana correcta v mediante el criterio

$$v = \underset{v}{\operatorname{argmax}}(x_k[n] - \gamma_k). \qquad k \in 2,3,\dots,15$$

Se están haciendo cumplir las restricciones para cada v

$$x_v[n] - \gamma_v \geq x_k[n] - \gamma_k. \qquad \forall\, k \in 2,3\dots..15 \qquad v \neq k$$

Por tanto, al utilizar el método con esta base de datos, se debería obtener un $\vec{\gamma}$ igual o muy similar al $\vec{\gamma}$ heurístico (ya que es el $\vec{\gamma}$ que permite cumplir las restricciones) y unas α_{vk}=0 (pues las restricciones han sido forzadas para que se cumplan siempre, sin tener en cuenta a $\vec{\alpha}$). Por tanto, se realizó esta prueba, obteniéndose un $\vec{\gamma}$ cuyos valores se asemejaban a los heurísticos, variando mínimamente como se puede observar en la

figura 11, y quedando los α_{vk} del orden de 10^{-10}. Esto se correspondía con lo esperado, y al comprobarse que el error $E_\varepsilon = 0$, se confirmó que el problema no estaba en el método ni en el programa, si no en la base de datos etiquetada.

Con lo cual se llegó a la conclusión de que, si se pudiera conseguir una base de datos suficientemente grande y que estuviera bien etiquetada, el método permitiría obtener el $\vec{\gamma}$ óptimo para cualquier registro.

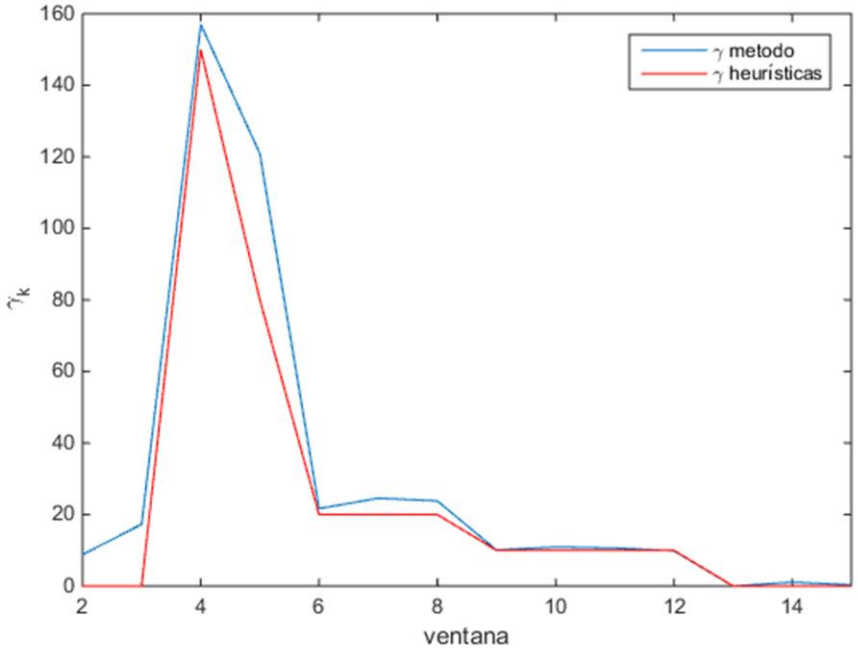

Figura 11. Comparación entre los γ_k heurísticos y los obtenidos mediante CVX

4.6 Robustez del método

El hecho de que con el etiquetado indirecto original se obtuvieran unos malos resultados hace necesario comprobar cómo afectan al método los errores en la etiquetación, a los que nos referiremos como *outlayers*. Por tanto, se planteó hacer un análisis del error cometido en función del porcentaje de *outlayers* introducidos. Lo que se busca con este análisis es ver a partir de qué porcentaje de *outlayers* el error se vuelve suficientemente alto como para decir que es inviable. Para este problema en concreto, un error medio de 1.5 se puede considerar aceptable, ya que el algoritmo se equivoque de media por mas ventanas implicaría que no está funcionando correctamente. En cuanto a la base de

datos para este análisis se utilizaron todos los registros de los que se disponía (41113), los cuales se etiquetaron de la misma forma que antes, usando el algoritmo diferencial ponderado con el $\vec{\gamma}$ heurístico. A partir de aquí el procedimiento es sencillo, siguiéndose los siguientes pasos:

- Se dispone de N_datos registros etiquetados.
- Se extraen N_test registros aleatorios para ser usados en nuestro método y obtener un $\vec{\gamma}$.
- Se introduce un porcentaje P de *outlayers* en las etiquetas de esos N_test datos. Estos *outlayers* son puramente aleatorios, siguiendo una distribución uniforme.
- Se calcula un $\vec{\gamma}$ mediante el método usando los N_test registros con el porcentaje P de *outlayers*.
- Se utiliza el algoritmo diferencial ponderado con el $\vec{\gamma}$ calculado en los N_datos registros originales, y se calcula el error medio.
- Se va incrementado el porcentaje P, obteniendo así la evolución del error a medida que los *outlayers* aumentan.
- Todo esto se realiza M veces para cada valor de P, calculando el error medio para cada uno de estos valores. Así se elimina la variabilidad debida la introducción de *outlayers* aleatorios –pues puede que haya coincidido que en una iteración los *outlayers* apenas se diferencien de las etiquetas originales, y en la siguiente haya diferencias de 8 y 9 ventanas.

En la siguiente figura se ven los resultados obtenidos para M=400, N_datos=41113, N_test=20000, y un P que varía entre 0 y 5% (es decir, se introducen de 0 a 1000 errores).

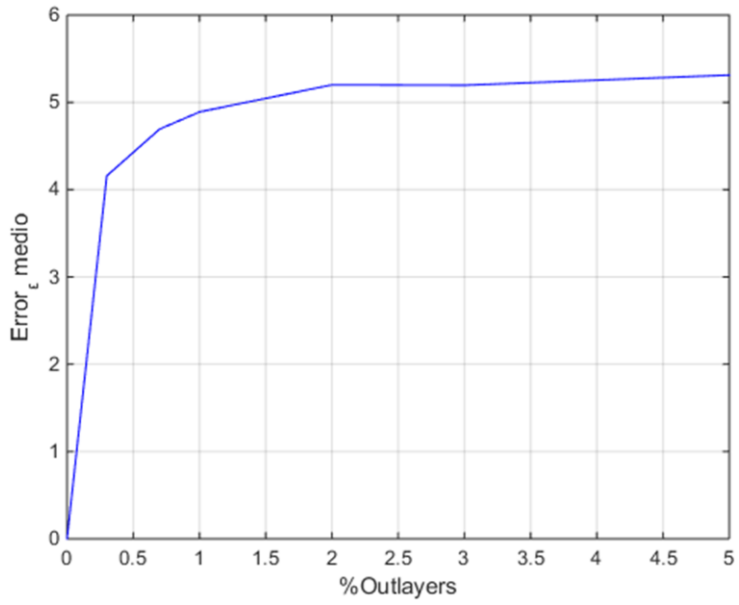

Figura 12. Evolución del error respecto al % de *outlayers*

Estos resultados muestran que hasta más o menos el 0.5% de *outlayers* el error crece linealmente a un ritmo muy rápido, y a partir de este valor empieza a estabilizarse, hasta llegar al 2%, donde se mantiene casi constante. Sin embargo, para estos porcentajes los valores de error ya son muy altos (equivocarse de media por 5 ventanas es equivocarse por mucho, ya que en total solo se tienen 15 ventanas). Debido a que, en general, los *outlayers* no supondrán porcentajes tan altos, y a que se quiere ver adecuadamente a partir de qué porcentaje el error supera el valor que se ha elegido como límite, en la figura 13 se muestran unos valores más centrados en la zona de interés.

Se puede comprobar que se comporta de manera prácticamente lineal, como se apreciaba en la figura anterior, y que el límite de error que se habían considerado, 1.5, se corresponde con un 0.05% de *outlayers*. Esto en número de errores son 10 errores para 20000 datos de test, lo cual hace ver que este método es bastante frágil frente a *outlayers*. Esta conclusión obliga a que, a la hora de utilizar este método, sea necesario estar muy seguro de que la base de datos está bien etiquetada y no incluye apenas errores.

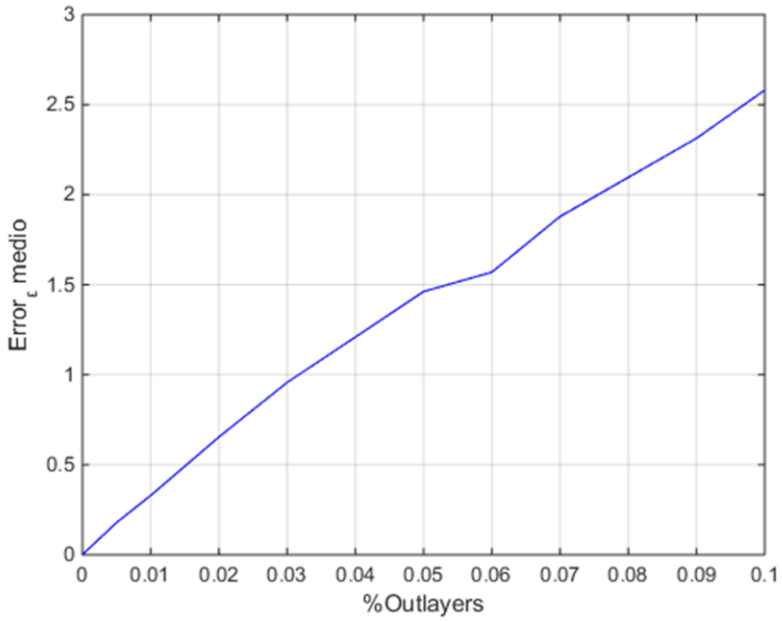

Figura 13. Evolución del error respecto al % de *outlayers*. Zona lineal

4.7 Solución final: Prestaciones

Por tanto, viendo el comportamiento del método que se ha implementado, se puede llegar a las siguientes conclusiones.

- Es objetivo: Una de las cuestiones en este problema es el hecho de que, aparte de la observación y estudio "a mano" de los registros de los que se dispone, no hay ningún otro método que permita obtener los parámetros de los algoritmos. Por tanto el poder evitar la subjetividad de una persona que analiza los registros es un gran avance, sobre todo al sustituirlo por un método que ya dispone de un criterio a seguir a la hora de obtener el parámetro $\vec{\gamma}$.

- Es automático: No solo se evita la subjetividad en la obtención del parámetro $\vec{\gamma}$, sino la necesidad de que haya una persona dedicada a él. Con esto también se elimina el problema de que, mientras que este método puede trabajar fácilmente con miles de registros, una persona se encuentra más limitada, pudiendo trabajar solo con unos cientos.

- Ahorra tiempo: Es notable mencionar el buen resultado del método respecto al tiempo de computo, pues usando un ordenador de bajas prestaciones el tiempo

necesario para procesar los 1025 registros es de unos 3 segundos, y el de los 41113 registros, poco menos de 10 segundos[3]. Esto permite evitar el tiempo que se debe dedicar al estudio de los registros, reduciéndolo a unos pocos segundos.

Sin embargo, hay que realzar el hecho de que es muy importante que la base de datos esté perfectamente etiquetada, pues ya se ha visto como unos pocos errores hacen que el error aumente en gran medida.

[3] En concreto, los tiempos de cómputo del método mencionados en este trabajo se corresponden con el tiempo requerido para obtener el parámetro $\vec{\gamma}$ en un Pentium 4 a 3GHz y con 2Gb de RAM.

5. Conclusiones y líneas futuras

En este trabajo se ha presentado un problema: la necesidad de obtener los niveles de llenado de una serie de contenedores haciendo uso del valor de potencia de un eco ultrasónico en 15 intervalos temporales. Aunque pudiera parecer sencillo lo cierto es que, tras enfrentarnos al alto número de variables que entran en juego, resulta complejo.

En un plano personal, este trabajo me ha servido personalmente para complementar y extender los conocimientos obtenidos a lo largo del grado. Como punto a destacar, el enfrentarse a un problema real me ha hecho ver que la realidad es bastante más compleja de lo que puede parecer a simple vista, escapándose en mayor o menor grado a cualquier intento de sistematización completa. Por otra parte, dado que ha sido necesario un uso en profundidad del programa Matlab, he mejorado sensiblemente mi grado de manejo sobre el mismo. Asimismo, he tenido que introducirme brevemente en el conocimiento de las SVMs, lo que me ha permitido ver una nueva herramienta y comprender su funcionamiento. El tener que usar la optimización convexa me ha servido para tomar conciencia de la relevancia y de la potencialidad de esta parte de la matemática, además de que me abre las puertas a un uso más profesional de la *toolbox* CVX de Matlab.

En cuanto a los resultados del trabajo, se puede decir que los objetivos buscados han sido cumplidos. Se ha establecido un criterio de comparación justo entre los algoritmos que había implementados en los sensores mediante el error *ε-insensitive*. Utilizando este criterio se concluyó que el algoritmo que mejor prestaciones ofrecía era el algoritmo diferencial ponderado, cuyo parámetro, el cual se denominó $\vec{\gamma}$, se había obtenido de manera heurística. Y, aunque en cuanto a la búsqueda de un parámetro óptimo no se han obtenido valores concretos, se puede afirmar haber conseguido un método que permite, haciendo uso de una base de datos adecuada, obtener los valores de dicho parámetro que minimicen el error.

En conclusión, la solución final que se presenta consiste en la utilización del algoritmo diferencial ponderado. Como parámetro a introducir, se utilizará un parámetro obtenido a partir del método que descrito en el capítulo 4, haciendo uso de una base de datos suficientemente amplia y sin errores. Esta solución aporta las siguientes prestaciones:

- Es objetiva: Como ya se comentó al final del capítulo 4, utilizar el método mencionado anteriormente permite eliminar la subjetividad derivada de tener a una persona etiquetando datos y eligiendo heurísticamente los valores del parámetro del algoritmo. Por tanto las decisiones que tome esta solución siempre se realizarán siguiendo el criterio de la base de datos introducida en el método.

- Es automática: Como ya se ha comentado en el punto anterior, esta solución permite hacer el cálculo del parámetro óptimo automático, eliminando la necesidad de que haya una persona buscando unos valores adecuados. Sin embargo, sigue siendo necesario que haya alguien dedicado a la etiquetación de una base de datos, la cual no debe apenas incluir errores.

- Minimiza el error: Se ha comprobado que, como mínimo, esta solución puede hacerlo igual con un parámetro calculado que con el heurístico del que se disponía. Esto indica que, si la base de datos que se utiliza está bien etiquetada, se asegura que el error a la hora de la decisión del nivel de llenado va a ser mínimo, al contrario que antes, donde al tener que usar soluciones heurísticas no se podía asegurar una solución ideal.

- Ahorra tiempo: Por todo lo anterior, se puede ver que esta solución permite ahorrar una gran cantidad de tiempo en la búsqueda heurística de parámetros y en la comprobación de registros. El hecho de que el método de obtención del parámetro óptimo sea rápido no hace más que abrir nuevas puertas a la hora de realizar pruebas, evitando perder el tiempo a la espera de los resultados de los cálculos.

Todos estas características no solo hacen ver que la solución obtenida es viable, si no que permiten plantear nuevas opciones para mejorar la optimización. El ejemplo más directo sería la extensión de esta solución para conocer el estado de llenado de contenedores de otro tipo, como vidrio o residuos orgánicos. En cuanto al primero, el hecho de que el vidrio forme una superficie relativamente uniforme (debido a que el recipiente se suele romper al caer) unido a que presenta un coeficiente de reflexión mayor que el cartón o el plástico hacen que esta solución sea casi directamente aplicable, e incluso presente mejores resultados. En cuanto a los desechos orgánicos, es

posible que su alta irregularidad supusiera un problema, pero probablemente no más que para el caso de los plásticos.

Otra idea que se podría aplicar es una calibración individual de cada sensor. Antes, al tener que realizar los análisis de los registros de manera manual, buscar soluciones particulares para cada sensor era impensable (se tienen miles de sensores por toda la ciudad, y calibrar cada uno de ellos analizando los registros requeriría un tiempo inmenso). Sin embargo, al disponer de un método automático que calcula el parámetro deseado partiendo unos datos etiquetados y en pocos segundos, la solución sería tan sencilla como pasar los registros etiquetados de cada sensor de manera iterativa, obteniendo el parámetro óptimo de cada sensor. Todo esto permitiría reducir en gran medida el error pues en vez de una solución genérica que tenga que tener en cuenta los problemas de todos los sensores, se tiene una solución individual para cada uno de ellos, y con un coste temporal mínimo.

BIBLIOGRAFÍA

[1] M. Mohri, A. Rostamizadeh and A. Talwalkar. *Foundations of Machine Learning*, The MIT Press, 2012.

[2] V.N. Vapnik. *The nature of Statistical Learning Theory*, Springer-Verlag, NewYork, 1999

[3] S.R. Gunn. *Support Vector Machines for Classification and Regression*, ISIS Technical Report ISIS-1-98, Image Speech & Intelligent Systems Research Group, University of Southapton, May. 1998

[4] J. Shawe-Taylor and N. Cristianini. *Kernel Methods for Pattern Analysis*, Cambridge University Press, New York, 2004

[5] Automotive Ultrasonic Sensor Interface for Park Assist or Blind Spot Detection Systems - Test Data. Texas instrument application note: TIDA-00151, 2014

[6] MATLAB Support Vector Machine Toolbox. http://www.isis.ecs.soton.ac.uk/resources/svminfo/

[7] S. Boyd and L. Vandenberghe. *Convex Optimization*, Cambridge University Press, New York, 2004

[8] Michael Grant and Stephen Boyd. CVX: Matlab software for disciplined convex programming, version 2.0 beta. http://cvxr.com/cvx, September 2013.

[9] Michael Grant and Stephen Boyd. Graph implementations for nonsmooth convex programs, Recent Advances in Learning and Control (a tribute to M. Vidyasagar), V. Blondel, S. Boyd, and H. Kimura, editors, pages 95-110, Lecture Notes in Control and Information Sciences, Springer, 2008. http://stanford.edu/~boyd/graph_dcp.html.