UNIVERSIDAD CARLOS III DE MADRID

ESCUELA POLITÉCNICA SUPERIOR

INGENIERÍA TÉCNICA TELECOMUNICACIÓN

SISTEMAS DE TELECOMUNICACIÓN

CLASIFICADOR ADAPTATIVO LINEAL PARA IGUALACIÓN DE CANAL

PROYECTO FIN DE CARRERA

por Alberto Ramos Sánchez de Pedro,

dirigido por Dr. Emilio Parrado Hernández,

codirigido por Dra. Matilde Sánchez Fernández

Marzo 2010

UNIVERSIDAD CARLOS III DE MADRID

ESCUELA POLITÉCNICA SUPERIOR

INGENIERÍA TÉCNICA TELECOMUNICACIÓN

SISTEMAS DE TELECOMUNICACIÓN

CLASIFICADOR ADAPTATIVO LINEAL PARA IGUALACIÓN DE CANAL

PROYECTO FIN DE CARRERA

AUTOR: Alberto Ramos Sánchez de Pedro

TUTOR: Dr. Emilio Parrado Hernández

COTUTOR: Dra. Matilde Sánchez Fernández

Marzo 2010

TÍTULO: Clasificador adaptativo lineal para igualación de canal

AUTOR: ALBERTO RAMOS SÁNCHEZ DE PEDRO

TUTOR: Dr. EMILIO PARRADO HERNÁNDEZ

COTUTOR: Dra. MATILDE SÁNCHZ FERNÁNDEZ

La defensa del presente Proyecto Fin de Carrera se realizó el día 17 de Marzo de 2010, siendo calificada por el tribunal

PRESIDENTE: Ascensión Gallardo Antolín

SECRETARIO: José Joaquín Escudero Garzás

VOCAL: Ricardo Vergaz Benito

Habiendo obtenido la siguiente calificación

CALIFICACIÓN

Presidente Secretario Vocal

A mis padres y hermano, por haberme ayudado, escuchado, aconsejado y apoyado en todos estos años, y especialmente en los peores momentos de esta aventura.

AGRADECIMIENTOS

Con la presentación de este proyecto y culminación de los estudios, llega el momento de agradecer a unas cuantas personas con las que tanto he compartido tanto el ámbito universitario como fuera de él, por haberme escuchado y ayudado en tantos momentos. Gracias a tod@s.

A Kant, Sara, Diego, Mario, Andrés, Quique, Chilio, mi primo David, Fany, Raquel, Carol, Poeta, Busta, David (Pelos), Borja, Maga, Fran, Iván, Vicente, a mi tutor Emilio y mi cotutora Matilde.

BREVE DESCRIPCIÓN

Con este proyecto hemos pretendido dar una solución alternativa a la igualación de canal, planteándolo como un problema de clasificación adaptativa, gracias a un algoritmo llamado *Tighter Budget Perceptron*, trabajando en un contexto de comunicaciones móviles mediante el uso de la tecnología HSDPA.

A pesar de que la solución del problema se plantea desde el punto de vista teórico, nos enorgullece haber sido capaces de plantear una solución distinta de la habitual para esta operación básica en el ámbito de las comunicaciones digitales (y móviles en nuestro caso), pues abre la posibilidad de que resolvamos otros problemas con los que nos enfrentamos a diario en telecomunicaciones con otros métodos y/o herramientas distintas de las utilizadas normalmente y de las que disponemos.

ABSTRACT

This project has been developed to provide an alternative solution to channel equalization, solving it as an adaptative classification problem thanks to an algorithm called *Tighter Budget Perceptron*, working in a mobile communications context using HSDPA technology.

Although the solution of the problem arises from the theoretical point of view, we are proud to have been able to pose different from the usual solution to this basic operation in digital communications (and mobile in our case), then opens the possibility for us to solve other problems that we face daily in telecommunications with other methods or tools other than those commonly used and available.

ÍNDICE GENERAL

1. Introducción……17

1.1. Marco Tecnológico……..22

1.2. Objetivos……...23

1.3. Estructura del Documento……..26

2. Tecnología Empleada en el Problema de Igualación de Canal en UMTS 28

2.1. Tecnología…….....29

2.1.1. UMTS……..29

- **Breve descripción de la arquitectura UMTS**……31

2.1.2. HSDPA……34

2.2. El Problema de Igualación de Canal en UMTS……...36

2.2.1. Parámetros de Estudio……..36

- **Rayos HSDPA**……..36
- **Ruido y pérdidas en el enlace radioeléctrico**…….37
- **Canales empleados en nuestro problema**……..39
- **Variaciones sufridas por la señal en canales móviles**……...40
- **Tipo de modulación**……43
- **Figura de mérito: BER**……44

2.2.2. Igualación de Canal en nuestro ámbito……...45

- **Aprendizaje máquina en el problema de igualación de canal**…47

3. Algoritmos de Aprendizaje Adaptativo 49

3.1. El Tratamiento de Datos y La Clasificación Máquina……..…….50

3.1.1. El Tratamiento de Datos……..…….50

3.1.2. La Clasificación máquina…….52

3.2. Clasificadores Lineales Un ejemplo: El Perceptrón.....55

3.2.1. Clasificadores Lineales…….…….....55

3.2.2. Un ejemplo de clasificador lineal: el Perceptrón……57

3.3. SVM y Clasificador Lineal de Máximo Margen……..61

3.3.1. Introducción a SVM……..61

3.3.2. Clasificador Lineal de Máximo Margen……..62

3.4. Caso No Separable…….....65

3.4.1. Introducción……...65
3.4.2. Optimización del número de errores……...66
3.5. Funciones de *Kernel*…….....68
3.5.1. Funcionamiento del *Kernel*……...69
3.5.2. *Kernel* empleado…….....70
3.5.3. Aplicación de la función *Kernel*……....70
3.6. Solución propuesta: *Tighter Budget Perceptron*……...73
3.6.1. Introducción……...73
3.6.2. Antecedentes: *Online Classification on a Budget*……73
3.6.3. *Tighter Budget Perceptron*…….75
4. Experimentos **79**
4.1. Descripción de los experimentos……...79
4.2. Resultados……80
4.2.1. Canal 1 Perceptrón con ventana……..81
4.2.2. Canal 1 SVM sin sesgo con ventana…….89
4.2.3. Canal 2 Perceptrón con ventana……..92
4.2.4. Canal 2 SVM sin sesgo con ventana…….98
4.2.5. Canal 3 Perceptrón con ventana……102
4.2.6. Canal 3 SVM sin sesgo con ventana……...104
4.2.7. Canal 1 Perceptrón sin ventana……105
4.2.8. Canal 1 SVM sin sesgo sin ventana……108
4.2.9. Canal 2 Perceptrón sin ventana……110
4.2.10. Canal 2 SVM sin sesgo sin ventana……112
4.2.11. Canal 3 Perceptrón sin ventana……113
4.2.12. Canal 3 SVM sin sesgo sin ventana……115
4.2.13. Resumen canales……..116
4.3. Guiado por decisión: comparación de resultados……..117
5. Conclusiones **121**
6. Presupuesto **124**
7. Referencias y bibliografía **125**

ÍNDICE DE FIGURAS

2.1.1 Técnica de espectro ensanchado…….31
2.1.2 Arquitectura de red UMTS…….33
2.2.1 Pérdidas en un enlace radioeléctrico……..38
2.2.2 Escenario Canal 1……...39
2.2.3 Escenario Canal 2……...40
2.2.4 Escenario Canal 3……...40
2.2.5 Representación efecto multitrayecto…….....42
2.2.6 Desplazamiento en frecuencia del efecto Doppler…….42
2.2.7 Constelación antipodal para señal BPSK……...44
2.2.8 Diagrama de bloques de nuestro problema…….48

3.1.1 Diagrama de bloques de un sistema de comunicaciones digitales…….51
3.1.2 Diagrama de bloques de la clasificación máquina……..52
3.1.3 Sobreajuste y subajuste……...54
3.2.1 Hiperplano de separación para dos conjunto de muestras……..56
3.2.2 Diagrama de bloques del Perceptrón monocapa “duro”…….57
3.2.3 Concepto de margen para un conjunto de muestras……...58
3.2.4 Concepto de margen geométrico para dos conjuntos de muestras…….59
3.2.5 Tipos de fronteras para dos conjuntos de muestras……60
3.3.1 Clasificador de margen máximo con vectores soporte……...65
3.4.1 Fronteras lineal y cuadrática con presencia de *outlier*……...67
3.5.1 Concepto de espacio transformado mediante *kernel*……..69
3.6.1 Ejemplo empleando *Tighter Budget Perceptron*……76

4.2.1.1 Gráfica resumen canal 1 Perceptrón con ventana……86
4.2.1.2 Gráfica comparativa canal 1 Perceptrón con ventana…….88
4.2.2.1 Gráfica resumen canal 1 SVM sin sesgo con ventana…….90
4.2.2.2 Gráfica comparativa canal 1 SVM sin sesgo con ventana……..92
4.2.3.1 Gráfica resumen canal 2 Perceptrón con ventana……95

4.2.3.2 Gráfica comparativa canal 2 Perceptrón con ventana…….97
4.2.4.1 Gráfica resumen canal 2 SVM sin sesgo con ventana……...100
4.2.4.2 Gráfica comparativa canal 2 SVM sin sesgo con ventana…….101
4.2.5.1 Gráfica resumen canal 3 Perceptrón con ventana……..104
4.2.7.1 Gráfica resumen canal 1 Perceptrón sin ventana……...107
4.2.7.2 Gráfica comparativa canal 1 Perceptrón sin ventana…….108
4.2.8.1 Gráfica resumen canal 1 SVM sin sesgo sin ventana……109
4.2.9.1 Gráfica resumen canal 2 Perceptrón sin ventana……...111
4.2.10.1 Gráfica resumen canal 2 SVM sin sesgo sin ventana……..112
4.2.11.1 Gráfica resumen canal 3 Perceptrón sin ventana…….114
4.2.12.1 Gráfica resumen canal 3 SVM sin sesgo sin ventana……..116
4.2.13.1 Gráfica comparativa entre los tres canales……..117
4.3.1 Comparación todo pilotos vs guiado por decisión canal 1……...119
4.3.2 Comparación todo pilotos vs guiado por decisión canal 2 Perceptrón…….120
4.3.3 Comparación todo pilotos vs guiado por decisión canal 3……...120

Tabla 1: Resultados Canal 1 Perceptrón con ventana 50 dB…….....82
Tabla 2: Mejores resultados Canal 1 Perceptrón con ventana…….86
Tabla 3: Resultados Canal 1 Perceptrón con ventana 10 dB……...87
Tabla 4: Resultados Canal 1 Perceptrón con ventana 0 dB…….88
Tabla 5: Resultados Canal 1 SVM sin sesgo con ventana 50 dB……89
Tabla 6: Mejores resultados Canal 1 SVM sin sesgo con ventana……..89
Tabla 7: Resultados Canal 1 SVM sin sesgo con ventana 30 dB……91
Tabla 8: Resultados Canal 1 SVM sin sesgo con ventana 10 dB……91
Tabla 9: Resultados Canal 2 Perceptrón con ventana 50 dB……...92
Tabla 10: Mejores resultados Canal 2 Perceptrón con ventana……...94
Tabla 11: Resultados Canal 2 Perceptrón con ventana 30 dB…….95
Tabla 12: Resultados Canal 2 Perceptrón con ventana 20 dB…….96
Tabla 13: Resultados Canal 2 Perceptrón con ventana 10 dB…….96
Tabla 14: Resultados Canal 2 Perceptrón con ventana 0 dB……...97
Tabla 15: Resultados Canal 2 SVM sin sesgo con ventana 50 dB……..98
Tabla 16: Mejores resultados Canal 2 SVM sin sesgo con ventana……99
Tabla 17: Resultados Canal 2 SVM sin sesgo con ventana 30 dB……100
Tabla 18: Resultados Canal 2 SVM sin sesgo con ventana 20 dB……100

Tabla 19: Resultados Canal 2 SVM sin sesgo con ventana 10 dB……101
Tabla 20: Resultados Canal 3 Perceptrón con ventana 50 dB……...102
Tabla 21: Mejores resultados Canal 3 Perceptrón con ventana…….104
Tabla 22: Resultados Canal 3 SVM sin sesgo con ventana 50 dB……105
Tabla 23: Resultados Canal 1 Perceptrón sin ventana 50 dB……106
Tabla 24: Mejores resultados Canal 1 Perceptrón sin ventana……..106
Tabla 25: Otros resultados Canal 1 Perceptrón sin ventana……..107
Tabla 26: Resultados Canal 1 SVM sin sesgo sin ventana……108
Tabla 27: Resultados Canal 2 Perceptrón sin ventana 50 dB……110
Tabla 28: Mejores resultados Canal 2 Perceptrón sin ventana……..111
Tabla 29: Mejores resultados Canal 2 SVM sin sesgo sin ventana……...112
Tabla 30: Resultados Canal 3 Perceptrón sin ventana 50 dB……113
Tabla 31: Mejores resultados Canal 3 Perceptrón sin ventana……..114
Tabla 32: Resultados Canal 3 SVM sin sesgo sin ventana……115
Tabla 33: Resumen mejores parámetros por canales…….116
Tabla 34: Comparación guiado por decisión vs todo pilotos…….....118

Notación empleada

N	dimensión del espacio de características
$\vec{y} \in Y$	salida y espacio de salida
$\vec{x} \in X$	entrada y espacio de entrada
F	espacio de características
$\langle \vec{x} \cdot \vec{z} \rangle$	producto escalar entre dos vectores
$\phi : X \to F$	proyección al espacio de características
$K(\vec{x}, \vec{z})$	*kernel* $\langle \phi(\vec{x}) \cdot \phi(\vec{z}) \rangle$
$f(\vec{x})$	función de valores reales
n	dimensión del espacio de entrada
l	tamaño del conjunto de entrenamiento
$\vec{w}$	vector de pesos
b	sesgo
$\vec{\alpha}$	multiplicadores de Lagrange o variables duales
L	forma primaria del Lagrangiano
W	forma secundaria del Lagrangiano
$\vec{x}', X'$	vector o matriz transpuesta
F(·)	función del decisor en clasificación máquina
R	números reales
S	conjunto de entrenamiento
η	parámetro de configuración de las funciones RBF
μ	umbral para clasificadores máquina
γ	margen
β	margen máximo
ξ	margen variable o *slack*
T	tamaño de la ventana deslizante

Capítulo 1
Introducción

En nuestro proyecto buscamos una solución alternativa a una operación básica en todas las transmisiones digitales, como es la igualación de canal, planteándolo como un problema de clasificación adaptativa, mediante un algoritmo llamado *Tighter Budget Perceptron* [1], trabajando en un contexto de comunicaciones móviles mediante el uso de la tecnología HSDPA [2] (*High Speed Downlink Packet Access*).

Descrito el problema a grandes rasgos y habiendo hecho mención a tres conceptos muy importantes, como igualación de canal, clasificación adaptativa y HSDPA, es conveniente desarrollarlos por separado antes de plantear la solución del problema.

Comencemos hablando de lo que es la igualación de canal. En comunicaciones digitales la igualación de canal no es más que un filtro receptor que tiene por objetivo compensar la distorsión introducida por el canal, para que a la salida el nivel de ISI [3] (*Inter-Symbol Interferente*, interferencia entre símbolos) sea menor que el presente en la señal de entrada [3].

Cuando se trabaja en comunicaciones de tipo radio, hay que considerar que la información transmitida hasta que llega al receptor atraviesa un medio concreto y se ve afectada por el ruido y una serie de pérdidas que existen en cualquier enlace radioeléctrico, como, por ejemplo, las introducidas por las antenas que transmiten la información o las de transmisión en el medio inalámbrico, entre muchas otras [4]. Dichas pérdidas corrompen la información en distinta medida según su intensidad, de manera que lo que se recibe no es lo mismo que se transmitió originalmente.

En nuestro caso concreto, en que suponemos un escenario de trabajo de comunicaciones móviles, además de las pérdidas que existen en el enlace, tenemos que considerar una serie de distorsiones propias de dichos canales que afectan a la señal,

como reflexiones, difracciones, dispersiones, desvanecimientos y efecto Doppler, provocado por el desplazamiento del móvil a altas velocidades.

La suma de todos estos elementos actúa sobre la señal transmitida "desfigurándola" y obliga al receptor a tratar de recuperar con la mayor fidelidad posible la información original. Para ello está la igualación de canal, que actúa a modo de filtro compensando los efectos de distorsión añadidos a la señal, corrigiéndolos y recuperando el mensaje lo mejor posible.

Existen distintos tipo de detectores: de secuencias, símbolo a símbolo, adaptativos, de mínimo error cuadrático medio o de retardo fraccionario, donde cada uno aplica distintas técnicas para recuperar la información. Para saber si la información que se recupera es correcta o no y para obtener una estima del canal que la mayoría de los detectores necesitan, se recurre a determinadas estrategias como el seguimiento por pilotos.

El seguimiento por pilotos consiste en que determinadas partes de la señal original se "marcan" para que sean visibles por el extremo receptor. Con ello, tenemos acceso al símbolo transmitido y podemos comparar el símbolo recuperado con el original.

Cuando no se tiene acceso al símbolo transmitido, se aplica la técnica de guiado por decisión [5] que consiste en comparar el signo del símbolo tratado con el predicho y ver si coinciden.

Sin embargo, la principal característica que nos interesa de los igualadores de canal es que se emplean en problemas como los de comunicaciones móviles, que poseen ciertas características muy importantes que deben ser tenidas en cuenta a la hora de plantear la solución. Nos referimos a [6]:

- procesamiento en tiempo real, las comunicaciones se realizan en determinados instantes que requieren respuestas inmediatas;
- existen limitaciones computacionales que no permiten la solución en bloque o *batch* (situaciones donde poseemos todos los datos desde el principio del problema[*]), ya que no se dispone de la información al completo y hay que analizarla a medida que se recibe;

[*] A diferencia de los algoritmos *online*, donde se reciben las muestras una a una, existe otra opción de trabajo, *batch*, donde se disponen todas las muestras desde el inicio del problema. Su mayor problema es el empleo de matrices muy grandes desde el comienzo, afectando a la carga computacional.

- nos encontramos en un entorno no estacionario donde se exige la necesidad de seguimiento ya que las características del canal, y por lo tanto del problema, varían con el tiempo.

Es decir, que implica una serie de retos a resolver y hacen el problema interesante, sobre todo, el procesamiento *online* de la información para seguir los cambios producidos.

Visto en qué consiste la igualación de canal, pasamos a hablar de la clasificación adaptativa.

Podemos empezar mencionando a dónde se aplica la clasificación máquina, pues se utiliza en ámbitos tales como la minería de datos, el reconocimiento de formas (por ejemplo, en medicina conteo de glóbulos blancos en una muestra de sangre) o la recuperación de información (como, extraer con el mayor acierto posible un tipo de información solicitada a un buscador de internet) [6].

La clasificación máquina es una parte del aprendizaje máquina, donde un algoritmo "aprende" a partir de una serie de ejemplos, para determinar si asigna una muestra a una u otra clase según sus características. Pensemos en el ejemplo anterior de contar glóbulos blancos en una muestra de sangre: dichas células tienen determinadas características que las diferencian del resto, de modo que el clasificador procesa todas las células que haya en la muestra determinando si cada una es o no un glóbulo blanco.

Así entonces, la clasificación máquina consiste en clasificar muestras de distintas clases mediante algún tipo de técnica o método. Podemos aplicar fronteras de tipo lineal, cuadrático, senoidal, cualquiera que resulte apropiada para resolver el problema. En un caso en que el problema sea linealmente separable, es decir, que aplicando una frontera lineal sea suficiente para resolverlo sin cometer error, no tendremos dudas de qué frontera aplicar, pero si no lo es deberemos determinar cuál nos interesa aplicar. Hay que considerar que cuanto más se ajuste la frontera a la solución óptima menos error se cometerá, pero a costa obtendremos una frontera más compleja que implica mayor coste computacional, además estaríamos sobreentrenando la máquina y perderíamos capacidad de generalización, es decir, que funcione para datos nuevos no tratados previamente [6].

Elegir el tipo de clasificador resulta un punto crítico en el diseño, ya que elegir uno u otro determinará la capacidad máxima de acierto que se puede lograr según el problema a tratar. Nosotros nos hemos fijado en los clasificadores de tipo lineal, dada su

facilidad de implementación y bajo coste computacional, aunque implique mayor tasa de error respecto otros clasificadores más sofisticados.

Dentro de los clasificadores lineales existen distintos tipos, y un caso concreto son los de máximo margen. Éstos sitúan el hiperplano en el punto donde las muestras más próximas de distintas clases quedan lo más lejos posible. Un ejemplo de clasificador de máximo margen, son las máquinas de vectores soporte [7] (SVM, *Support Vector Machines*).

Por hiperplano nos referimos a una superficie que nos permita separar muestras que pertenecen a distintas clases, como son un punto en una dimensión, una recta en dos dimensiones, un plano en tres, y un hiperplano en un espacio mayor de tres dimensiones.

Las SVM son interesantes porque actúan como clasificadores lineales en espacios muy complejos minimizando el error sobre datos no tratados previamente. Esto es así porque a la hora de establecer el hiperplano de separación emplean un reducido número de muestras, en lugar de trabajar sobre todo el conjunto. Ese reducido grupo son aquellas de distintas clases más próximas entre sí, que reciben el nombre de vectores soporte [7].

El problema de los clasificadores de máximo margen es que se requiere que el problema sea linealmente separable, lo cual dadas las características de nuestro problema, no se garantiza. Para corregir esta limitación, donde el concepto de margen carece de sentido en situaciones no separables linealmente, las SVM emplean funciones de *kernel* que transforman un problema no lineal en otro que sí lo es, aplicando métodos semilineales evaluando un producto escalar en un espacio proyectado mediante una función en un espacio inicial [7].

Habitualmente el ámbito de funcionamiento de las SVM es en entornos estacionarios, donde no varía la física del problema y la solución se encuentra analizando todas las muestras varias veces para encontrar el mejor hiperplano posible.

Sin embargo, últimamente se han desarrollado algoritmos que trabajan en escenarios *online* que optan por una clasificación adaptativa, en función de una nueva medida de calidad (el número de errores cometidos en la clasificación), según las circunstancias en lugar de situar el hiperplano a partir de un margen geométrico, y que encaja muy bien en nuestro problema. Uno de estos algoritmos es el ya presentado *Tighter Budget Perceptron*, que emplea conjuntamente los principios de clasificadores adaptativos en situaciones *online* y métodos *kernel*, al que le haremos las

modificaciones pertinentes para adaptarlo a las características de nuestro escenario de simulación de un problema de comunicaciones móviles.

Y finalmente, terminamos este bloque comentando brevemente qué es HSDPA. HSDPA es una tecnología de reciente aplicación utilizada como optimización de la tecnología espectral WCDMA [8] (*Wideband Code Division Multiple Access*), que se emplea en el estándar UMTS [9] (*Universal Mobile Telecommunications System*) de la telefonía móvil de tercera generación (3G), mejorando diversos aspectos de funcionamiento e incrementando las velocidades de descarga de datos respecto WCDMA, además de permitir que cualquier dispositivo móvil se convierta en un terminal con acceso a la red de banda ancha.

Una vez que hemos presentado cada uno de los conceptos por separado, entendemos mejor cómo se ensamblan los tres conceptos entre sí para resolver el problema propuesto: replanteamos el problema de igualación de canal, que se produce en entornos de comunicaciones móviles (aquí entra en juego el uso de la tecnología HSDPA, definiéndonos los canales) como un problema de clasificación adaptativa (pues el entorno de trabajo es *online* y hay que seguir los cambios que se producen en el canal, como ocurre en las comunicaciones móviles) empleando clasificadores lineales adaptativos por medio de un algoritmo llamado *Tighter Budget Perceptron*.

Obviamente la mejor forma de resolver el problema consistiría en diseñar alguno de los igualadores de canal empleados para tal fin, como, por ejemplo, el algoritmo de Viterbi [3] utilizado en GSM [10] (*Global System for Mobile Communications*), que es el estándar utilizado en los teléfonos móviles de segunda generación (2G). Pero no es objeto de nuestro estudio competir con los igualadores de canal empleados en la telefonía móvil actualmente. Y dado que este problema ya está resuelto, desde nuestro punto de vista nos ha parecido más interesante y original enfocar el proyecto resolviendo un problema presente en las comunicaciones móviles, como es la igualación de canal, desde una perspectiva de simulación teórica y averiguar si el algoritmo posee habilidad suficiente para actuar como un igualador de canal encontrando los parámetros que conducen al mínimo error.

1.1. Marco Tecnológico

Planteada la resolución del problema no tratándolo como un problema de comunicaciones digitales, donde tendríamos que diseñar elementos como el codificador, el modulador y el demodulador entre otros, sino como otro de clasificación donde únicamente tenemos unas entradas y unas salidas para resolverlo, nuestro interés se centra en el diseño del igualador que se ajuste a las características requeridas.

Como el papel de igualador lo hace un clasificador lineal, empezaremos por saber qué es un clasificador lineal, veremos algún ejemplo, e iremos incorporando mayor complejidad hasta llegar al modelo final adoptado.

Comenzamos estudiando el problema de clasificación suponiendo que las muestras son linealmente separables, para lo cual el Perceptrón [11] es un buen punto de partida puesto que se ideó como clasificador lineal y ha servido de base para el desarrollo de clasificadores más complejos. Una vez analizado y vistos sus puntos débiles, pasamos a otro tipo de clasificador lineal, el de máximo margen, que estudiamos como un caso concreto de SVM.

El último paso para el diseño del clasificador será su aplicación en espacios linealmente no separables, estudiando el uso de las funciones *kernel* y los clasificadores que nos permiten construir, hasta presentar algunos de los algoritmos empleados en la clasificación en situaciones *online*, como *Tighter Budget Perceptron*, analizando por qué es el modelo escogido para simular un igualador de canal.

En cuanto a HSDPA, como se trata de una tecnología de reciente aplicación que persigue conectar cualquier dispositivo portátil con conexión inalámbrica (como agendas personales, teléfonos móviles, consolas o reproductores de música) a la red de banda ancha utilizando las redes de telefonía ya instaladas, parece propicio utilizarlo para recrear nuestros canales de trabajo y simular con el mayor realismo posible un entorno de comunicaciones móviles de tercera generación.

Dado que HSDPA mejora a su antecesor WCDMA, nos parece interesante empezar presentando los orígenes de la telefonía móvil 3G, describiendo brevemente su funcionamiento, características, mejoras y servicios ofrecidos respecto su antecesora en la telefonía móvil, GSM. De esta forma, tendremos una visión más completa del

problema, viendo todos los detalles que cabría considerar, y que no realizamos por no ser objeto de estudio de este proyecto. A medida que vayamos presentando y desgranando las distintas partes, iremos acotando el campo de acción limitando el problema.

Además el uso de HSDPA para simular los canales de trabajo confiere al problema cierto "realismo", ya que las especificaciones que empleamos para los canales se recogen en 3GPP *release* 5 (*3rd Generation Partnership Protocol*) [12]. Con ello, evitamos utilizar canales artificiales creados por nosotros mismos o de dudosa procedencia, que podrían ajustarse mejor a nuestros propósitos.

En definitiva, el problema resulta interesante porque unimos dos conceptos en principio separados: la igualación de canal en comunicaciones móviles, y los clasificadores lineales empleados habitualmente en problemas de reconocimiento de patrones o recuperación de información en situaciones estacionarias. El gran reto pasa porque, como hemos detallado, dos de los grandes problemas de las comunicaciones móviles residen en la velocidad de desplazamiento (que ocasiona efecto Doppler) y el multitrayecto de la señal que recibe el terminal y que provoca efecto ISI. La misión de los igualadores es corregir tal problema y recuperar la información sin dicho efecto. Y en nuestro caso tratamos de averiguar si transformando el problema en otro de clasificación, conseguimos que el algoritmo se comporte como un igualador de canal siguiendo los cambios producidos en el canal, por medio de una baja tasa de error.

1.2. Objetivos

Para encontrar la mejor tasa de error posible en nuestras simulaciones, tendremos que manipular una serie de parámetros que especificamos a continuación.

Empecemos conociendo la señal que manejamos: creada aleatoriamente de tipo binario, con valores +1 y -1, que se ve distorsionada por el ruido aditivo blanco

gaussiano AWGN (*Additive White Gaussian Noise*) [13] presente en telecomunicaciones, el efecto Doppler y el multitrayecto provocado por atravesar uno de los tres canales HSDPA que hemos supuesto.

Así entonces, los símbolos o bits que forman la señal llegan a nuestro igualador y tendrán cualquier valor real (por limitaciones de nuestro algoritmo no trabajamos con valores complejos), según les haya afectado el canal y el ruido, y son con los que trabajemos para recuperar la información.

Para averiguar si los símbolos recuperados son o no correctos, empleamos la técnica antes mencionada usada por los igualadores de canal: seguimiento por pilotos.

El modo de funcionamiento de nuestro algoritmo consiste en calcular el valor de cada símbolo utilizando, además del actual, *n* símbolos previos. Así averiguamos cuánto influye el pasado reciente sobre cálculos actuales en situaciones *online*. La corrección se produce cada vez que hay un error o cuando el valor calculado para una muestra es inferior a un parámetro β. En cuanto al método matemático con el que obtenemos o *predecimos* cada símbolo, se recurre a la forma del *kernel* gaussiano [7].

Puesto que las condiciones del canal varían continuamente, la forma de calcular las *predicciones* tienen en cuenta esta característica. Por ello, tendremos que averiguar cuántos símbolos previos influyen en los cálculos del símbolo *enésimo* dadas las características de los canales estudiados, por ejemplo símbolos de cincuenta o doscientos instantes previos, pues es lógico suponer que no influirán positivamente en los cálculos, complicándolos y además se necesitarían grandes cantidades de memoria.

Determinar cuántos símbolos usamos para los cálculos resulta un aspecto fundamental en nuestro diseño, ya que averiguar el número óptimo que induzca al mínimo error, es fundamental para la buena resolución del problema.

En cuanto a los símbolos que empleamos para la *predicción*, son los previamente presentados como vectores soporte, puesto que son los críticos para clasificar al resto. Para saber si un símbolo es o no vector soporte, tiene que incumplir la restricción del parámetro β, antes descrito. El conjunto de vectores soporte se almacenan en una caché o array de valor finito, evitando con ello que crezcan indefinidamente aumentando la complejidad de los cálculos a realizar y la cantidad de memoria requerida.

Mencionado el hecho de que, dado las características de nuestro problema, lo indicado para realizar las *predicciones* es utilizar sólo los vectores soporte más recientes

despreciando al resto, supone un nuevo reto: averiguar qué tamaño de la caché es el más indicado para resolver el problema.

Otro problema surge cuando la caché está llena y se tiene que añadir otro, entonces se debe elegir qué vector soporte se elimina. *Tighter Budget Perceptron* plantea que sea el que deje la máquina con menor tasa de error, almacenando los más críticos del diseño. Pero debemos saber sobre qué símbolos se calcula ese error, para lo que *Tighter Budget Perceptron* define una caché secundaria y dos estrategias: guarda todos los símbolos analizados hasta ese momento o los vistos hasta ese instante más cercanos al margen.

Dado que nuestro problema es no separable, sustituimos el concepto de margen por una ventana con T símbolos anteriores al actual (comprobando la influencia del pasado reciente en situaciones *online*), y en cuanto al vector soporte que sacamos de la caché, probamos dos casos: "eliminamos" el que obtenga menor error sobre los datos de nuestra ventana o borramos el vector soporte más "antiguo".

Puesto que el problema es *online*, desconocemos los vectores soporte verdaderos que resuelven el problema de manera óptima, así que tenemos que ajustarnos a las circunstancias de cada momento quedándonos con los que son los mejores para nosotros, es decir, que ocasionan menor error.

A la hora de realizar los cálculos, cada vector soporte se multiplica por una constante, α, que ayuda en la *predicción* de cada símbolo según las ecuaciones definidas por dos reglas distintas: Perceptrón y SVM sin sesgo [14], añadiendo un nuevo parámetro en el problema.

Hasta ahora hemos mencionado cuatro parámetros con los que tendremos que "jugar" para resolver el problema: el tamaño de la caché, cuántos vectores soporte utilizamos en las predicciones, cuántos símbolos anteriores toma cada vector soporte y la técnica empleada para amplificar o disminuir el *peso* de cada vector soporte. Sigamos viendo parámetros.

Las pruebas se realizan en tres escenarios distintos con dos parámetros de configuración: la velocidad a la que se desplaza el móvil y la cantidad de rayos que atraviesan cada canal y llegan al terminal o estación móvil, abreviado como MS [15] (*Mobile Station*). Será interesante comprobar la relación entre la tasa de error obtenida y dichos parámetros, para determinar cuánto influyen en nuestro problema para seguir los cambios que se producen en el canal, lo cual analizaremos con las pruebas realizadas.

El valor de SNR [16] (*Signal to Noise Ratio*, Relación Señal a Ruido) también es otro parámetro a estudiar, para averiguar cuánto influye en el resultado final según la cantidad de ruido presente en el sistema. Puesto que trabajamos en canales móviles, la relación SNR que utilizaremos será positiva en dBs, ya que es un requisito fundamental de dichas comunicaciones si queremos que la calidad sea aceptable. De lo contrario, las condiciones de funcionamiento están tan degradadas que la comunicación no podría realizarse y nuestro programa encontraría grandes problemas para trabajar en dicha situación.

1.3. Estructura del Documento

Una vez presentado el problema y la solución propuesta, detallamos cómo se estructura el proyecto.

El capítulo 2 se divide en dos bloques, el primero tratará sobre la principal tecnología implicada en nuestro problema, HSDPA. Puesto que es un protocolo aplicable a la telefonía móvil 3G, comenzamos hablando y detallando su tecnología base, UMTS, para entender mejor HSDPA. El segundo bloque versa sobre el problema de igualación de canal a estudiar, particularizado a nuestra situación, y analizando los elementos necesarios para su completa explicación, como canales empleados, ruido existente, tipo de señal, variaciones que sufre la señal, modulación empleada y medida de calidad elegida para obtener conclusiones de los experimentos.

El capítulo 3 tiene por objetivo detallar los aspectos de la clasificación adaptativa. Para ello, comenzamos planteando el problema de clasificación máquina, centrándonos en los clasificadores de tipo lineal en situaciones en las que las muestras son linealmente separables, y a continuación los clasificadores de máximo margen. Para explicar su construcción y funcionamiento acudimos a las máquinas de vectores soporte (SVM), y con ello presentar algunos elementos interesantes que nos son útiles como los vectores soporte. Una vez visto esto y qué inconvenientes presentan en nuestro caso,

analizamos otro tipo de clasificador donde las muestras no son separables linealmente, que nos sirve para presentar las funciones *kernel*. Finalmente, hablamos de algunos algoritmos empleados actualmente en escenarios *online* que combinan principios de las SVM y métodos *kernel*, para llegar al adoptado como solución final, *Tighter Budget Perceptron*.

El capítulo 4 está dedicado a la parte experimental, donde detallamos las pruebas realizadas, los resultados obtenidos con tablas y gráficas, y su consiguiente discusión para determinar si resultan o no óptimos desde nuestras consideraciones iniciales según los parámetros escogidos.

En el capítulo 5 recapitulamos las conclusiones obtenidas comparando las posiciones inicial (partiendo de lo esperado teóricamente) y final (con los resultados en la mano).

El capítulo 6 incorpora la parte bibliográfica y referencias que nos han ayudado a completar la información a lo largo del documento.

Por último, el capítulo 7 se encarga de completar el proyecto incluyendo el presupuesto teórico que ha conllevado el estudio: material de oficina y tecnológico empleado, software, alquiler de equipos o línea telefónica.

Capítulo 2

Tecnología empleada en el Problema de Igualación de Canal en UMTS

En el capítulo segundo abordamos la resolución del problema de igualación de canal particularizado en un escenario UMTS en dos bloques.

Primero tratamos la parte tecnológica, empezando describiendo la tecnología UMTS empleada actualmente en la telefonía móvil 3G, para presentar una tecnología más moderna, HSDPA, empleada no sólo en telefonía 3G, y ver qué ventajas aporta respecto UMTS.

Además, HSDPA nos ayuda a definir el medio en el que simulamos las pruebas realizadas para resolver nuestro problema de la forma más real posible en cuanto al escenario simulado.

En el segundo bloque, planteamos completamente el problema de igualación de canal y su solución, exponiendo en primer lugar todos los elementos necesarios para definirlo, por medio de: modulación de la señal empleada, canales utilizados, tipo de ruido existente, variaciones que sufre la señal o figura de mérito escogida para evaluar nuestras pruebas.

Finalmente, conocidos todos los parámetros los integramos en la resolución del problema de igualación de canal junto con el algoritmo propuesto.

2.1. Tecnología

2.1.1 UMTS

UMTS supone la evolución de GSM y es una tecnología estándar empleada, entre otras, en la telefonía móvil de tercera generación. Vista la evolución que ha tenido la telefonía móvil desde su popularización a finales de los años 90, podemos suponer que UMTS surge como respuesta a la necesidad de ofrecer nuevos y mejores servicios respecto a los ofertados por GSM para satisfacer las demandas de los usuarios y ampliar el campo de negocio de la telefonía móvil.

De las muchas diferencias entre ambas tecnologías, mencionamos como una de las más importantes la que posibilita nuevos servicios, a causa del incremento de velocidad binaria en UMTS gracias a la tecnología WCDMA [8] (*Wideband Code Division Multiple Access*, Acceso Múltiple por División de Código de Banda Ancha), que es la tecnología móvil inalámbrica de tercera generación que permite mayor velocidad de transmisión de datos para dispositivos móviles. Este incremento de velocidad se debe al hecho del mayor ancho de banda de los canales en UMTS, 5 MHz frente a los 200 KHz utilizados en GSM.

Otros datos importantes de WCDMA que nos serán útiles para la realización de los experimentos son, la duración de la trama de 10 mseg que incluye 15 *slots* o ranuras de tiempo, teniendo cada *slot* 2560 chips, lo que da un total de 38400 chips por trama. El ancho de banda de la señal UMTS viene dada por la velocidad de 3.84 Mchips/seg, y será la tasa de simulación que empleemos en nuestro sistema [8].

WCDMA admite una velocidad de transmisión de datos que puede oscilar entre 144–512 Kbps en entornos de cobertura amplia, y los 2 Mbps teóricos para áreas locales. Ya sabemos que las velocidades teóricas nunca coinciden del todo con las alcanzadas en situaciones reales por diversos motivos y, en este caso, el amplio rango de la velocidad se debe al número de usuarios conectados a la red en una misma zona en el mismo instante, que hace que haya que repartir el espectro entre todos ellos [17].

Poder alcanzar hasta 2 Mbps (o velocidades cercanas, pero siempre superiores a las ofertadas por GSM) se traduce en:

- nuevas aplicaciones de banda ancha y capacidades multimedia, como son la transmisión de audio y vídeo en tiempo real y videoconferencia;
- mayor calidad de voz, hasta poder equipararla a la de la red telefónica fija;
- posibilidad de soportar el protocolo IP [18] (*Internet Protocol*),

A estas capacidades descritas hay que añadir otra diferencia respecto GSM, y es que permite realizar distintas comunicaciones simultáneas como, por ejemplo, transmitir ficheros y mantener una videoconferencia simultáneamente.

En cuanto a la seguridad, si en GSM las comunicaciones de voz y datos estaban cifrados y existía un sistema de autentificación complejo para acceder al sistema por parte de los terminales y así impedir la lectura de las comunicaciones, en UMTS el empleo de distintas formas de multiplexación o control de acceso al medio como CDMA [8] (*Code Division Multiple Access*, Acceso Múltiple por División de Código) combinado con FDD [19] (*Frequency Division Duplex*, Multiplexación por División en Frecuencia) y TDMA [19] (*Time Division Multiple Access*, Acceso Múltiple por División en el Tiempo) nos permite que en una misma frecuencia cada usuario transmita con un código distinto incrementando notablemente la seguridad.

CDMA se basa en la tecnología de espectro ensanchado donde se transmite en la misma frecuencia con distinto código permitiendo aumentar la velocidad de transmisión de datos. Uno de los motivos para emplear la tecnología de espectro ensanchado CDMA se debe a que protege las comunicaciones, ya que al ensanchar la frecuencia de la señal queda por debajo del nivel de ruido, y lo convierte en imposible de detectar. **Figura 2.1.1.**

FDD se usa en los canales de subida y bajada entre el móvil y la antena y viceversa.

TDMA distribuye unidades de información en ranuras o *slots* alternas de tiempo proporcionando acceso múltiple a un reducido número de frecuencias.

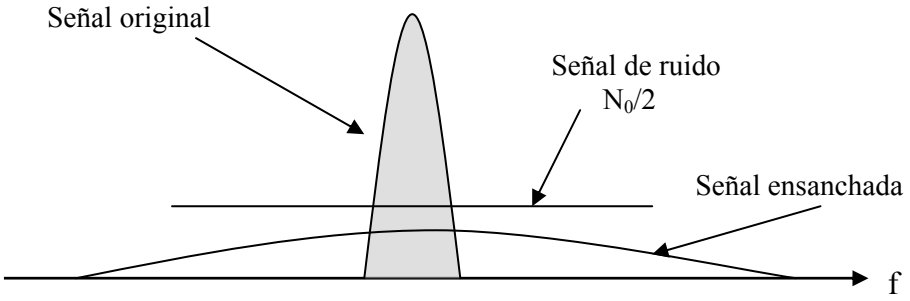

Figura 2.1.1 Efecto de aplicar la técnica de espectro ensanchado sobre una señal

Pero no todo son ventajas en UMTS ya que, como toda tecnología, también tiene puntos débiles, entre los que podemos mencionar:

- la limitación de la cobertura en algunas zonas donde las compañías de telefonía móvil todavía no proveen de acceso a los móviles 3G;
- la disminución de la velocidad si el dispositivo móvil desde el que nos conectamos está en movimiento (por ejemplo si circulamos en automóvil);
- la elevada latencia, referida al retardo producido en la transmisión y propagación de los paquetes en la red que puede afectar a determinadas aplicaciones novedosas que ofrece UMTS como los juegos en línea, que requieren de una conexión cliente-servidor.

▪ Breve descripción de la arquitectura en UMTS

A pesar de que en nuestro problema no perseguimos la planificación de cobertura que deben cubrir las antenas ni de la arquitectura de red UMTS, consideramos conveniente describir determinados elementos de su arquitectura para poder presentar algunas características interesantes de la tecnología HSDPA que sí utilizamos en el problema.

Dentro de los múltiples componentes que constituyen la arquitectura de la telefonía móvil 3G, nos centramos en dos: la Red de Acceso Radio Terrestre UMTS denominada UTRAN [8] (*UMTS Terrestrial Radio Access Network*) y el propio MS. En la **Figura 2.1.2** se muestra un esquema de red UMTS simplificado [8].

Al diseñar UMTS se intentó aprovechar toda la infraestructura desplegada para la telefonía móvil 2G, en cuanto a la red de antenas y protocolos de funcionamiento, pero eso sí, incorporando el *hardware* y *software* necesarios que permitieran desarrollar los nuevos servicios pretendidos por UMTS, como es el caso de la red UTRAN.

UTRAN corresponde a una parte de la arquitectura de red UMTS, rediseñada respecto a GSM, que permite altas velocidades de transmisión y se encarga de proporcionar la conexión entre el terminal móvil y el Núcleo de Red o CN (*Core Network*) [4]. Éste consta de varios elementos encargados de la conmutación, señalización y encaminamiento de la información desde y hacia redes externas, como son la red telefónica pública conmutada (PSTN, *Public Switched Telephone Network*) [15], redes IP y la red digital de servicios integrados (ISDN, *Integrated Services Digital Network*) [15].

UTRAN se compone de una serie de Subsistemas de Redes de Radio denominado RNS [8] (*Radio Network Subsystem*), que a su vez engloban al Controlador de la Red de Radio o RNC [8] (*Radio Network Controller*) y uno o varios nodos B [9].

El RNS gestiona la asignación y liberación de recursos radio para permitir la comunicación con los MSs en un área determinada.

El RNC es la entidad controladora de un RNS y se encarga de tareas como el control general de los recursos radio proporcionados por uno o varios nodos B, de las decisiones de *handover* que requieren señalización al MS, y ofrece conexión al CN. *Handover* es el término utilizado en comunicaciones móviles para referirse a las transferencias de servicio entre nodos B cuando la calidad de la comunicación ofrecida al MS se degrada por diversas circunstancias.

Los nodos B son los responsables de la transmisión/recepción radio hacia/desde los MSs en una o más celdas UMTS, y gobiernan una o varias antenas.

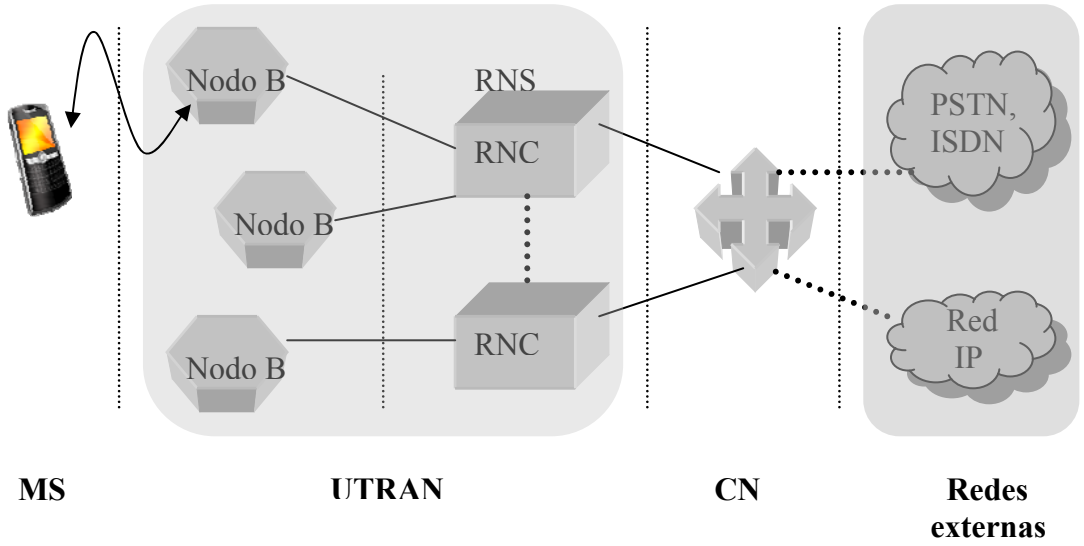

Figura 2.1.2 Esquema reducido de arquitectura de red en UMTS, donde las líneas discontinuas muestran distintos tipos de interfaces existentes. [4]

A la extensión de terreno que cubrimos con una o más antenas agrupadas se denomina celda, y a su vez cada nodo B puede dar cobertura a una o más celdas. Esto constituye otra diferencia entre UMTS y GSM, ya que en GSM las unidades encargadas de ofrecer la cobertura, denominadas BSS [20] (*Base Station Subsystem*, Subsistema de Estaciones Base), a los MSs lo hacen sólo en una única celda. El hecho de que en UMTS podamos conectarnos a más de una antena ofrece por un lado la ventaja de reforzar las comunicaciones, y por otro el inconveniente de que se consumen más recursos del sistema. Al proceso de conectarse a más de una antena se llama *soft-handover*.

Existe otro concepto relacionado con el traspaso entre distintas celdas de cobertura sin perder servicio, llamado *hard handover*, que de momento no tratamos.

A la hora de seleccionar la ubicación de las antenas existen diversos factores como intentar maximizar la cobertura o el tráfico que se pueda cursar, pero también nos enfrentamos a limitaciones en cuanto a su emplazamiento: obstáculos en el medio, permisos de instalación o resistencia por parte de los vecinos a su ubicación. Nosotros estudiamos si la mayor o menor presencia de obstáculos en el medio y sus consecuencias en la cantidad de rayos HSDPA que recibe nuestro MS, afecta a la resolución del problema, sin preocuparnos por modelar la ubicación de las antenas.

En lo referente al MS, más que centrarnos en los componentes que lo forman, nos interesa saber que en las pruebas realizadas se desplaza a altas velocidades y recibe las tramas de información que analizamos para resolver el problema planteado.

2.1.2. HSDPA

Ahora detallamos la tecnología HSDPA empleada en los canales para simular nuestras pruebas, cuyas especificaciones técnicas ya avanzamos que están disponibles en 3GPP *release* 5.

HSDPA es una tecnología móvil de reciente aplicación con el objetivo de mejorar la tecnología actual UMTS. De hecho, a HSDPA se la conoce como 3.5G, es decir, que servirá de transición entre la actual 3G y la futura cuarta generación de telefonía móvil (4G).

Como mencionamos al presentar UMTS, HSDPA se diseña de manera que los nuevos protocolos sean compatibles con su predecesor para así incorporar las mejoras y nuevas aplicaciones de manera más sencilla. En la telefonía 3G, HSDPA aprovecha toda la infraestructura ya desplegada con UMTS, al igual que ésta hacía con GSM. Obviamente, HSDPA comporta una serie de cambios, tanto en la adición de nuevos elementos en la arquitectura de red UMTS como la implementación de nuevos protocolos de funcionamiento, para lograr su fin, pero que no serán objetos de estudio en nuestro problema puesto que no perseguimos su implementación.

HSDPA se diseñó para conseguir un mayor ancho de banda respecto UMTS/WCDMA gracias a un nuevo canal compartido entre todos los usuarios en el enlace descendente, de ahí el apellido *Downlink* en HSDPA. El nuevo canal mejora tanto la eficiencia del espectro como la velocidad de transferencia de datos hasta los 14 Mbps, consiguiendo tasas de transmisión promedio en torno a 1 Mbps. Comparados con los actuales 2 Mbps *teóricos* que ofrece como máximo UMTS supone un avance sustancioso. Es decir, HSDPA incrementa la eficiencia espectral respecto WCDMA, de la misma forma que éste la aumentaba respecto GSM o GPRS (*General Packet Radio Services*) [21].

El hecho de incrementar la eficiencia espectral y la velocidad, comparándolo con WCDMA, se traduce en la aplicación de nuevos servicios, aplicaciones y que un mayor número de usuarios emplee la red simultáneamente, ya que también se disminuye la latencia por debajo de los 100 ms (que será importante para desarrollar algunas aplicaciones). Otra de las ventajas, es que mientras que en WCDMA el ancho de banda se distribuye entre los usuarios conectados en ese momento, HSDPA proporciona entre tres y cuatro veces más de capacidad permitiendo que más usuarios compartan el canal con buenas prestaciones de servicio.

HSDPA además de mantener intactas las ventajas que aportaba WCDMA respecto GSM, incorpora, gracias a la mayor eficiencia espectral, un nuevo concepto: *cell breathing* (respiración celular). Ésta afecta a la planificación del tamaño de las celdas UMTS, que hace que no sean constantes, pues les hace adaptarse a las circunstancias según el número de usuarios y servicios que se requieran en un determinado momento, de forma similar a si crecieran o encogieran de tamaño.

El incremento de velocidad y disminución de la latencia aportadas por HSDPA, se traducen en la disminución del tiempo de conexión y mejora de la calidad de las aplicaciones en tiempo real (videoconferencia, juegos *online* entre múltiples jugadores). En cuanto al resto de servicios que ofrece UMTS, como la navegación web, descarga de archivos o lectura de correo electrónico, con HSDPA sucede lo mismo que en los casos anteriores: se permite mayor velocidad de conexión a todos estos servicios empleando menor tiempo.

HSDPA alcanza sus elevadas tasas de velocidad y ofrece más capacidad gracias a técnicas tales como la modulación empleada 16-QAM [22] (*Quadrature Amplitude Modulation*, Modulación de Amplitud en Cuadratura de 16 estados), la codificación variable de errores y la redundancia incremental implementado con la técnica HARQ [23] (*Hybrid Automatic Repeat Request*, Petición Automática de Respuesta Híbrida). [24]

Actualmente se trabaja en la mejora de este protocolo, conocida como HSDPA+, que persigue mejorar las velocidades de transmisión descendentes de los 14 Mbps actuales hasta los 42 Mbps. Las fases segunda y tercera de este protocolo, ya están en marcha y encaminadas a aumentar la tasa de transmisión de datos empleando nuevas técnicas de multiplexación, como OFDM (Multiplexación por División de Frecuencias

Ortogonales, *Orthogonal Frequency Division Multiplexing*) [25], y antenas inteligentes tipo MIMO (Múltiple entrada múltiple salida, *Multiple-Input Multiple-Output*) [26].

Sin meternos en un estudio profundo de funcionamiento del protocolo HSDPA, nos quedamos con la idea de que su objetivo es permitir la conexión de banda ancha, asociada hasta ahora con un equipo fijo (como un ordenador de sobremesa conectado a la banda ancha), de un dispositivo móvil genérico en cualquier situación y momento, aprovechando las redes establecidas de telefonía móvil con las frecuencias ya asignadas y siendo completamente compatible con la tecnología actual para aprovechar las ventajas y servicios ofrecidos.

2.2. El Problema de Igualación de Canal en UMTS

2.2.1. Parámetros de Estudio

- **Rayos HSDPA**

El número de rayos HSDPA se refiere a los rayos que recibe nuestro terminal móvil procedentes del nodo B que nos ofrece cobertura en un determinado instante, que varía continuamente debido a múltiples causas: movimiento del móvil, desvanecimientos de la señal o presencia de obstáculos en el medio.

En concreto lo que nos interesa es averiguar cómo influye este parámetro en la resolución del problema suponiendo tres casos distintos con 2, 4 y 6 rayos HSDPA, pues con ello estudiamos los efectos del multitrayecto (cuando la señal procede de distintos caminos) en nuestro estudio.

Características tales como la ubicación y orientación de las antenas o *soft-handover* no se estudian en el problema, ya que los canales que empleamos no contemplan su manipulación.

▪ Ruido y Pérdidas en el enlace radioeléctrico

Un elemento muy importante a la hora de diseñar cualquier sistema de comunicaciones es tratar el ruido presente en el sistema o medio, ya que se trata de un elemento perturbador que corrompe la información a transmitir.

Además de las componentes frecuenciales (que desconocemos, puesto que de lo contrario no tendría sentido filtrar ni igualar) que añada el canal y distorsionan nuestra señal de trabajo, el ruido supuesto en el sistema es el presente habitualmente en telecomunicaciones, como es el ruido aditivo blanco gaussiano, AWGN. Es decir, que se suma a nuestra señal de interés, de tipo gaussiano aleatorio y caracterizado porque sus valores de señal en dos instantes de tiempo distintos no guardan correlación estadística.

Dada la propia naturaleza del ruido AWGN, que añade a nuestra señal componentes aleatorias, nos causa un problema más ya que no podemos filtrarlo ni tratarlo puesto que desconocemos sus componentes de frecuencia, y por lo tanto no sabemos que tipo de filtrado lo elimina. Para filtrarlo con plenas garantías tendríamos que conocer sus componentes frecuenciales, y eso implica saber cómo se comporta el canal. Pero saber la forma que tiene el canal supone que no haya problema de igualación de canal que resolver, puesto que sabríamos la transformación que sufre la señal de entrada.

En lo referente a las pérdidas en el enlace radioeléctrico, nos referimos a las pérdidas sufridas por la señal transmitida entre la antena transmisora y la receptora del MS (conocido en telefonía móvil como enlace descendente) o viceversa, del MS a la antena (llamado enlace ascendente).

El tipo de pérdidas a las que estamos expuestos son:

- pérdidas propias del enlace (L_g), que se definen como el cociente entre la potencia transmitida y la potencia entregada al receptor;

- pérdidas básicas de las propias antenas (L_b), puesto que toda la potencia suministrada a los dispositivos no se entrega a la antena (se producen pérdidas a causa de los conductores y los cables, entre otros);
- pérdidas de transmisión (L_{tx}), definidas como el cociente entre la potencia entregada por el transmisor y la potencia disponible en el receptor;
- pérdidas básicas de transmisión en espacio libre (L_{bf}), definidas como $\left(\frac{4\pi d}{\lambda}\right)^2$ donde d indica la distancia entre ambas antenas, y λ es la longitud de onda.

En la **Figura 2.2.1** se muestra cada una de las pérdidas descritas.

Todas estas pérdidas supondrían nuevos cálculos para incluir en nuestro proyecto, pero puesto que no es objeto de nuestro estudio ver cómo afectan dichas pérdidas en el problema expuesto, para facilitar cálculos se ha supuesto un escenario ideal en el que el único elemento, de los descritos, que perturbe la señal será el tradicional ruido AWGN al que otorgaremos distintas potencias de ruido para comprobar cómo influyen en el resultado final.

Cabe recordar que la potencia de ruido que añadimos hace que la relación SNR sea positiva en dBs (la potencia de la señal es mayor que la del ruido presente), lo cual, ya indicamos, es un requisito imprescindible de los canales telefónicos para obtener buena calidad de las comunicaciones.

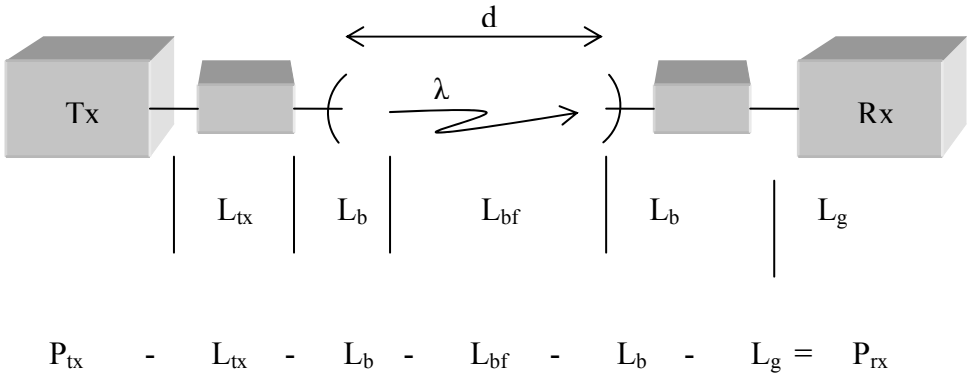

Figura 2.2.1 Resumen de pérdidas en un enlace radioeléctrico

▪ Canales empleados en nuestro problema

Nuestro escenario de trabajo se desarrolla en un medio evaluando tres canales con distintas características recogidos en 3GPP *release* 5. De esta manera, evaluamos el comportamiento del problema en distintos ambientes y cómo influyen en el resultado final.

Las características de cada canal son:

Canal 1: velocidad del MS 120 Km/h, 6 rayos HSDPA. Esta situación parece propia de un circuito urbano con calles estrechas y edificios altos, que impiden bastante la visión directa entre la antena y el MS. De manera que, la altura de los edificios y la anchura de las calles influyen de manera notable provocando un mayor número de rebotes de la señal hasta que llega al terminal móvil.

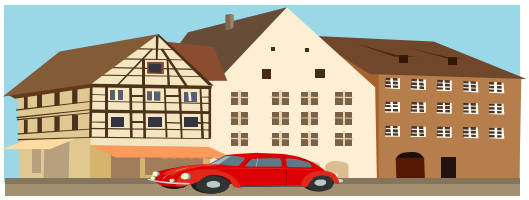

Figura 2.2.2 Escenario posible para el Canal 1: recorrido por ciudad

Canal 2: velocidad del MS 120 Km/h con 4 rayos HSDPA. Para este escenario, podría suponerse la hipotética situación de atravesar alguna pequeña zona montañosa o encontrarnos en circuito no urbano con algunos obstáculos que dificultan en parte la visión directa entre la antena y el MS, originando de nuevo multitrayecto en la señal recibida.

Figura 2.2.3 Posible situación para el Canal 2: túnel en carretera urbana

Canal 3: velocidad del MS 50 Km/h, 2 rayos HSDPA, lo que indica una situación con muy buena visibilidad entre la antena transmisora y nuestro dispositivo móvil sin apenas obstáculos, como puede corresponder a una autovía situada en terreno llano.

Figura 2.2.4 Situación para Canal 3: autovía

- **Variaciones sufridas por la señal en canales móviles**

Una vez definidos los tres canales al completo, es el momento de unir los conceptos de cobertura entre la antena y el MS con la velocidad de desplazamiento del propio terminal para explicar la relación entre ambos.

A medida que el MS se desplaza por el terreno, entra y sale de la zona de cobertura (celdas) que ofrecen las distintas antenas de la red. A este proceso de traspaso entre celdas en telefonía móvil se conoce como *hard handover*.

Visto que en nuestros escenarios suponemos velocidades elevadas del MS, parece lógico pensar que el proceso de *hard handover* se producirá en varias ocasiones.

Cada vez que el MS sale de una celda y entra en otra, los nodos B deben dar cuenta de dicho cambio producido, para tener en todo momento localizado al MS en el sistema por si necesitase enviar o recibir información.

Al igual que con el concepto de s*oft-handover*, *hard handover* tampoco se estudiará porque los canales utilizados no lo contemplan, pero parecía conveniente presentarlo y aclarar que no lo simularemos.

Así que, además de las pérdidas del enlace radioeléctrico antes descritas, existen otras causadas por la propagación existente en canales móviles, como son [4]:

- reflexiones (multitrayecto), que se pueden producir debido a la presencia de obstáculos entre la antena y el MS (edificios, por ejemplo) provocando que la señal transmitida proceda de distintos caminos produciendo una distorsión en frecuencia y efecto ISI;
- difracciones, producidas por ejemplo a causa de la diferencia de altura entre la antena transmisora (situada en un valle) y el MS (situado en una zona de terreno más elevada);
- variación Doppler debido a la velocidad del MS, que ocasiona una variación del nivel de la señal recibida y provoca que la frecuencia portadora se vea desplazada según la velocidad del MS una cantidad determinada de frecuencia.

La **Figura 2.2.5** representa el efecto multitrayecto y la **Figura 2.2.6** el efecto Doppler.

Para paliar el efecto Doppler que sufrirá nuestra señal, UMTS aprovecha una de las ventajas de WCDMA al ensanchar el espectro y ofrecer mayor eficiencia espectral, puesto que soporta mejor diversos tipos de errores. Tal y como vimos en la **Figura 2.1.1** al ensanchar la señal, dura más tiempo y se disminuyen los efectos que provoca la interferencia, respecto los que ocasionaría sobre la señal original (que es de menor duración). Es decir, se consigue mayor protección frente a errores o ruido.

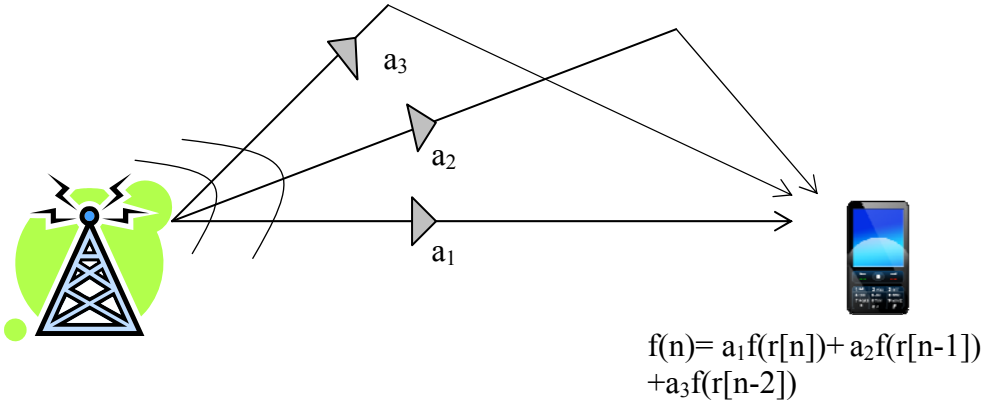

Figura 2.2.5 Representación del efecto multitrayecto en el instante 'n', donde recibimos el componente de la señal en ese instante, además de dos instantes anteriores. Cada término va multiplicado por una constante, que es la potencia de señal.

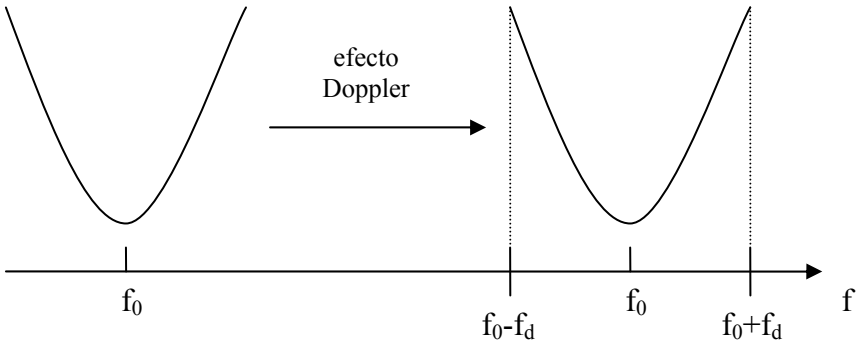

Figura 2.2.6 Desplazamiento de la frecuencia portadora a causa del efecto Doppler

Las consecuencias del multitrayecto son sin duda uno de los grandes problemas de las comunicaciones móviles, por los efectos de ISI que ocasiona, y además sufrimos efecto Doppler por el desplazamiento a altas velocidades, por lo que los canales empleados contemplan estos fenómenos para estudiarlos.

Es por ello, que de entre todas las pérdidas descritas a las que podemos vernos sometidos, sólo tratamos el efecto Doppler y el multitrayecto, para comprobar cuánto nos afectan, y no complicar en exceso la resolución del problema. Para atenuar el efecto Doppler, se incorporan filtros específicos integrados en nuestra solución [27].

El *software* con el que simulamos las consecuencias del efecto Doppler y el multitrayecto de cada canal sobre los datos de entrada, está diseñado de forma que cada trama de datos tiene la longitud de una trama en UMTS, es decir, de 38400 chips, que en nuestro caso serán símbolos binarios aleatorios. Simularemos tres tramas de datos para realizar los experimentos, actuando sobre un conjunto total de $115.2 \cdot 10^3$ símbolos.

▪ Tipo de modulación empleada

El tipo de modulación de la señal empleada es BPSK [28] (*Binary Phase Shift Keying*), que es una modulación de diferencia de fase binaria perteneciente a la clase de modulación lineal paso banda angular, consistente en hacer variar la fase de la portadora entre un número de valores discretos, en este caso binario.

El empleo de este tipo de modulación es ideal para nuestros experimentos, puesto que BPSK se basa en que a cada símbolo le corresponde la misma potencia, que en nuestro caso es el número de rayos HSDPA disponibles en cada canal de forma constante (2, 4, ó 6) durante las pruebas realizadas.

La modulación paso banda es aquella que concentra la energía en torno a una frecuencia determinada y no respecto a la frecuencia cero. Es la modulación habitual empleada al trabajar en radiocomunicaciones debido a que empleamos distintas frecuencias para distintos servicios. En el problema, empleamos la frecuencia de trabajo de telefonía móvil 3G situada en torno a los 2 GHz, en concreto 2.025 GHz para la señal portadora (aquella que transporta la señal a transmitir).

El tipo de modulación (angular) empleada PSK es la que modifica la fase de la señal (incluyendo un desfase o adelanto). Y en concreto, la modulación BPSK se caracteriza porque sus posibles símbolos son +1 y -1 (binaria), por lo que presenta una constelación antipodal, tal y como se representa en la **Figura 2.2.7**: símbolos con la misma amplitud pero con fase contraria (0º y 180º), de ahí el nombre de diferencia de fase.

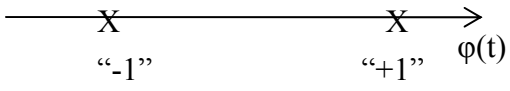

Figura 2.2.7 Constelación antipodal para una señal con modulación BPSK

Al emplear una señal binaria, no tenemos que integrar en nuestro sistema un decodificador de símbolos para saber que símbolo se transmitió, puesto que con un solo bit es suficiente para decodificar el símbolo.

- ## Figura de mérito: BER

Tal y como hemos enfocado el problema, la mejor forma que tenemos de averiguar si el algoritmo es capaz de funcionar es calculando la BER (*Bit Error Rate*), que calcula el número de bits erróneos o incorrectamente recibidos respecto al número total de bits enviados [13], para distintos parámetros y niveles de SNR que supongamos.

Una menor tasa de error de bit indicará que el algoritmo empleado es capaz de seguir los cambios que se producen en el canal. Si por el contrario, un experimento finaliza con altas tasas de error indicará que el algoritmo probado con esos parámetros no es apropiado para resolver el problema ya que le *cuesta* más seguir los cambios producidos en el canal, traduciéndose en mayores tasas de error.

Finalizados los experimentos, sólo tendremos que fijarnos en los que obtengan menor BER para saber qué parámetros escogidos son los más apropiados en cada situación para resolver nuestro problema.

2.2.2. Igualación de Canal en nuestro ámbito

Una vez que hemos presentado todos los parámetros a considerar en nuestro problema, procedemos a exponer nuestro particular problema de igualación de canal.

La idea que nos puede servir como punto de partida es que cuando se transmite información por cualquier tipo de red de comunicaciones, se debe manipular para tratar de recuperarla con el mayor acierto posible. En nuestro caso, una videoconferencia, una llamada telefónica o una descarga de datos pueden requerir la transmisión de gran cantidad de información en tiempo real a través de la red UMTS, donde no recuperar los paquetes de datos produce un deterioro de las comunicaciones.

Las limitaciones a las que nos enfrentamos para transmitir toda esa información por las redes son el ancho de banda (rango de frecuencias sobre el que se concentra la mayor parte de la potencia de una señal [29]) y el ruido. Del ruido ya hemos hablado, sobre los tipos de perturbaciones que nos afectan y cómo los solucionamos en nuestro caso, y sabiendo que sólo se puede trabajar en un ancho de banda concreto por una serie de limitaciones (físicas, administrativas o económicas), nos centramos en resolver el problema de cómo recuperar la información transmitida en tiempo real: mediante la igualación de canal.

La igualación de canal consiste en diseñar un filtro que al pasar la información recibida por él, devuelve una estimación de la información transmitida originalmente, compensando los efectos distorsionantes que haya incluido el canal y reducir así el nivel de ISI. Nuestro filtro realiza técnicas adaptativas (puesto que los canales son variantes en el tiempo), mediante el algoritmo *Tighter Budget Perceptron*, para realizar la estimación de la información.

Con la siguiente explicación y su consiguiente representación mediante la **Figura 2.2.8** detallamos el proceso completo de nuestra simulación.

El MS, como ya indicamos, recibe las tramas de información binarias (BPSK) formadas por $115.2 \cdot 10^3$ símbolos de valor +1 ó -1, creando la señal de entrada s[n], con las características de una señal UMTS, modulada en una señal portadora de frecuencia 2.025 GHz y transmitida a una tasa de $3.84 \cdot 10^6$ chips/seg [4].

El planteamiento que seguimos es que dicha señal atraviesa uno de los canales con características HSDPA estudiados, que añaden a s[n] la distorsión provocada por el efecto Doppler a causa del multitrayecto que sufre la señal. Para facilitar el planteamiento, a continuación aplicamos el filtrado Doppler para tratar tal variación y "limpiar" la señal todo lo posible. Con ello, una vez filtrada, resulta más fácil añadirle el ruido AWGN antes de trabajar con ella en el igualador, puesto que sólo hay que establecer la potencia de SNR que deseemos para definir r[n]. Es decir, en dos pasos consecutivos añadimos el efecto Doppler y su filtrado a s[n], y en último lugar le añadimos la potencia de ruido que corresponda antes de trabajar con ella en el algoritmo.

La función de transferencia de los filtros de Doppler viene dada por la función $h(t) = 2^{1/4}(2\pi f_d t)^{1/4}\Gamma(3/4)J_{1/4}(2\pi f_d t)$ [30], siendo $f_d = v/\lambda$ [4], donde *v* representa la velocidad del MS. Una vez que la señal pasa por el filtrado Doppler, se ve corrompida por cierta componente de ruido AWGN, e[n], por medio de la SNR definida, lo que provoca que cada bit que recibamos vea afectado su valor de nuevo y creándose la señal r[n]. Es decir, si el bit transmitido fue +1, las componentes de ruido añadidas tanto por el efecto Doppler como por AWGN, provocan que se reciba cualquier otro valor real (según cuánto y cómo le hayan afectado). Ya explicamos anteriormente que no podemos tratar las componentes de ruido AWGN mediante filtrado al desconocerlas.

Finalmente, procesamos símbolo a símbolo la señal r[n] comparando el valor predicho con su valor original por medio del seguimiento por pilotos.

Como los símbolos originales son binarios, y nuestras *predicciones* son valores reales, para compararlos necesitamos una función cuantificadora (la función signo) que aplicamos a r[n], produciendo una salida y[n] formada por símbolos de valor +1 ó -1. Sin más que restar uno a uno los valores de s[n] e y[n], sabemos si se ha cometido o no un error y si hay que corregir el algoritmo.

Detallado el problema de igualación de canal, damos una breve explicación sobre el funcionamiento del algoritmo de aprendizaje máquina que usamos para resolverlo.

▪ Aprendizaje máquina en el problema de igualación de canal

El algoritmo se modifica en dos situaciones posibles: cuando se produce un error (como acabamos de ver anteriormente) o cuando el valor de *predicción* es inferior a un parámetro β, según las ecuaciones definidas por *Tighter Budget Perceptron*. Si la *predicción* para un símbolo es correcta y no inferior a β, pasamos al siguiente símbolo. En cambio, los símbolos cuyo valor de *predicción* sea incorrecto o inferior a β se denominan vectores soporte, se almacenan en un array de longitud finita y representan al conjunto de símbolos críticos en el diseño del clasificador para el resto de símbolos.

Los vectores soporte no son más que vectores que guardan además del valor del símbolo actual recibido, *n* valores previos, creando así un vector de *n*+1 símbolos. Se utilizan para calcular la *predicción* del resto de símbolos gracias a las ecuaciones definidas por las funciones de *kernel* gaussiano, que se detallarán cuando tengamos una visión más completa del problema. Tendremos que probar cuántos valores previos son idóneos para resolver el problema.

Cada vez que se incorpora un símbolo a dicho array se rehacen los cálculos de *predicción*, y es en la gestión de ese array finito donde surge un nuevo problema: saber qué vectores soporte se deben considerar del total y omitir los demás. Encontrar el límite de vectores soporte y saber cuáles recordar para realizar los cálculos de las *predicciones*, serán dos puntos críticos en el diseño del clasificador, pues tratamos de encontrar la importancia del pasado reciente en un problema *online* donde los canales varían continuamente.

De esta manera se procede para todos los símbolos que conformen la secuencia de datos. Según los valores que asignemos a los parámetros en nuestro programa, los valores de BER variarán y conoceremos qué valores conducen al error mínimo.

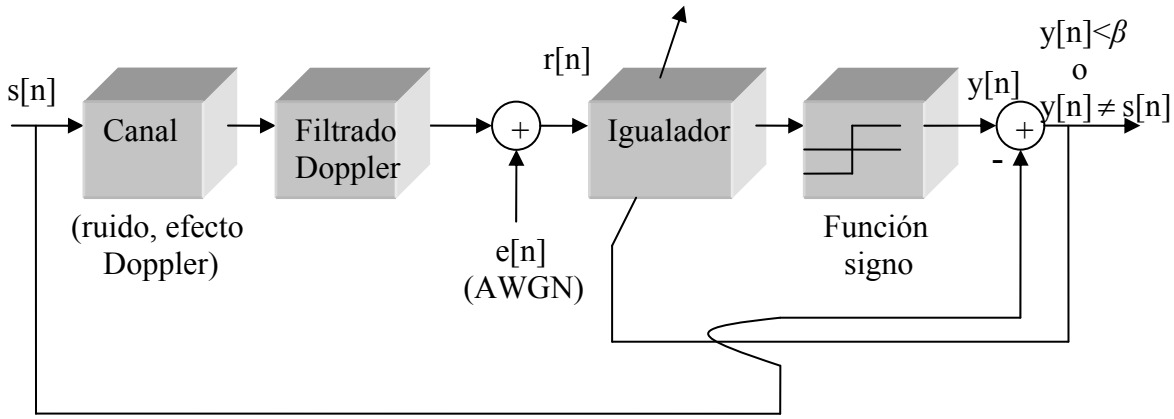

Figura 2.2.8 Diagrama de bloques de nuestro problema

Capítulo 3

Algoritmos de Aprendizaje Adaptativo

En el tercer capítulo presentamos el problema de la clasificación máquina adaptativa para situaciones *online*.

Comenzamos por lo que es el tratamiento de datos en comunicaciones, para ayudarnos a definir la clasificación máquina y su resolución mediante clasificadores lineales. Una vez tratado el caso más sencillo, describimos el clasificador lineal de máximo margen que emplea otra técnica para separar muestras en escenarios linealmente separables.

A continuación, tratamos el caso especial cuando se manejan muestras no separables linealmente, viendo otro tipo de clasificadores según los errores cometidos y cómo y cuándo nos ayudan las funciones de *kernel* para resolver situaciones más complejas.

Por último, llegamos a los algoritmos que conjugan características de todos los clasificadores tratados, por medio del que adoptamos como solución, *Tighter Budget Perceptron*, para "simular" el filtro del igualador de canal y resolver el problema en un escenario adaptativo UMTS.

3.1 El Tratamiento de Datos y la Clasificación Máquina

3.1.1 El Tratamiento de Datos

El tratamiento de datos es una constante en telecomunicaciones, ya que desde que una fuente de información (ordenador, teléfono) transmite un mensaje por cualquier canal y medio posible (conductores, infrarrojos o inalámbricos) hasta que llega al receptor, el mensaje ha sufrido distintas técnicas y procesos, para tratarlo, adaptarlo y modificarlo para así recibirlo de la mejor forma posible. Es decir, se han tenido que tratar los datos en cada uno de los procesos que sufre el mensaje. Algunos de los pasos que sigue la información en un sistema de comunicaciones digitales, representados en la **Figura 3.1.1** [28], se detallan a continuación:

- codificación, para convertirlo de una señal analógica a una digital, eliminar la redundancia posible del mensaje y añadir protección frente a los errores del canal;
- encriptación, para evitar que "extraños" accedan al mensaje, buscando con ello privacidad y autentificación;
- multiplexación, combina información de distintas fuentes o características para enviar el mensaje por el mismo sistema de comunicación digital, es decir, divide las señales en el medio por el que se propaguen;
- modulación, adapta la información a transmitir a las características del canal mediante una onda portadora que siga las variaciones de la onda moduladora (información a transmitir);
- transmisión y recepción, con los dispositivos adecuados (cables o antenas) para enviar y recibir la información;
- demodulación, recupera la información contenida en la onda portadora;
- demultiplexación, recupera las fuentes de información en un único medio de transmisión;

- desencriptación, emplea los códigos apropiados para leer el mensaje;
- decodificación, elimina la redundancia que contenga el mensaje y lo transforma al formato de la señal original.

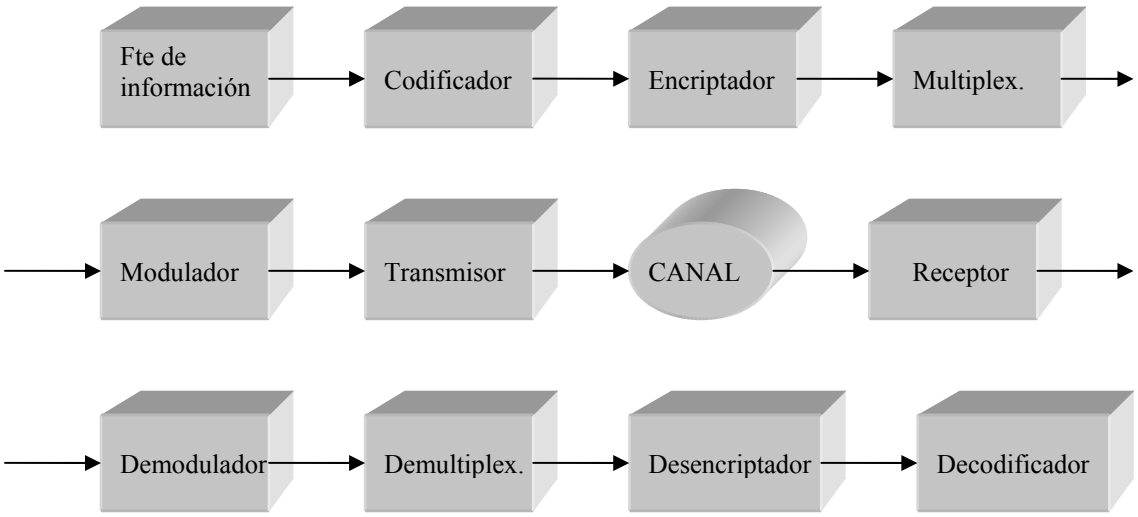

Figura 3.1.1 Diagrama de bloques elemental de un sistema de comunicaciones digitales

Ya sea para la realización de una llamada entre teléfonos móviles, el envío de un *email* o la compresión de un archivo informático, los datos enviados han "sufrido" estos y otros procesos (no descritos y específicos de cada acción).

Dentro de los distintos tipos de tratamiento de datos encontramos un caso particular, la clasificación máquina, que tratamos a continuación para ayudarnos a presentar conceptos e ideas interesantes, desgranar y resolver nuestro problema de igualación en canales UMTS.

3.1.2 La Clasificación Máquina

El término de clasificación máquina se refiere al proceso por el cual una máquina (según una arquitectura concreta) emplea un algoritmo (procedimientos para entrenar y clasificar) concreto, toma un conjunto de observaciones como entradas y a partir de ellas produce una salida etiquetada concreta.

Para visualizar mejor este proceso, presentamos un ejemplo clásico de clasificación máquina para un caso binario, cuyo diagrama de bloques se representa en la **Figura 3.1.2** [6]: un cliente, denotado $\vec{x}$, solicita a su banco la concesión de un préstamo, y el banco tendrá que decidir en función de un criterio concreto (decisor $F(\cdot)$) si concede o no el crédito solicitado (salida discreta D_1 ó D_0).

En este caso, el criterio escogido por el banco para tomar su decisión consiste en realizar una serie de preguntas al cliente y evaluarlas. Las hipótesis de entrada posibles que pueden darse, son que el cliente sea moroso (H_0) o no moroso (H_1) y los valores de salida conceder el préstamo (D_1) o denegarlo (D_0).

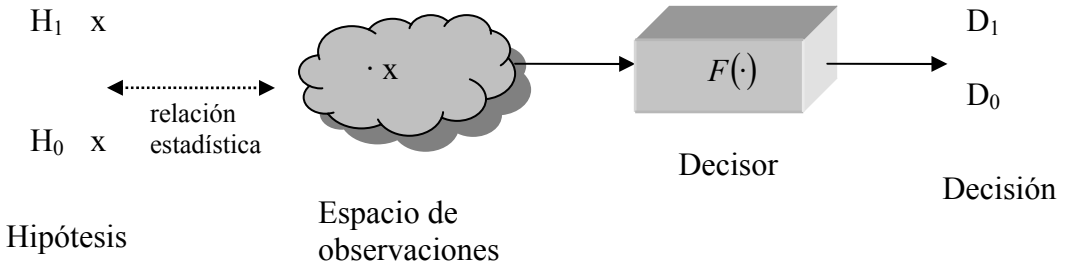

Figura 3.1.2 Visión de la clasificación máquina

De entre las muchas cuestiones posibles planteadas por el banco para tomar su decisión, encontramos: importe del crédito, patrimonio del cliente, si se encuentra trabajando o desempleado, cantidad de la nómina o fin por lo que se solicita el préstamo. Una vez que el cliente ($\vec{x}$) responde $(x_1, x_2, ..., x_n)$ a estas preguntas, ya

tenemos nuestras variables $\vec{x} = (x_1, x_2, ..., x_n)$ para tomarlas como las entradas de la máquina o decisor, que aplica la función $F(\cdot)$ para evaluarlas y ofrecer una salida.

$F(\cdot)$ puede adoptar cualquier forma (lineal, cuadrática, senoidal, exponencial, etc) y en el caso lineal o semilineal corresponderá a una combinación de pesos $\vec{w}$ concreto. Por ejemplo, en caso lineal sería $F_{\vec{w}}(\vec{x}) = w_0 + w_1 x_1 ... + w_n w x_n$, o cuadrático $F_{\vec{w}}(\vec{x}) = w_0 + w_1 x_1 + w_1^{'} x_1^2 + ... + w_n^{'} x_n^2$. Escogida la forma de $F(\cdot)$, cada pregunta tendrá un *peso* o importancia distinta, w_i, que ayuda a obtener el resultado final. Puede que el banco considere que encontrarse trabajando sea más importante que su patrimonio, por lo que dicha pregunta tendrá mayor peso.

La forma en que el decisor proporciona una salida para cada cliente, se debe al hecho de que anteriormente se le ha "entrenado" con datos de solicitantes previos de los que se conoce la respuesta correcta (muestras etiquetadas sabiendo si el cliente es o no moroso). De manera que cuando se le introducen los datos de un nuevo solicitante, la máquina tiene cierto "conocimiento" previo sobre otros casos tratados con los que comparar el nuevo caso.

Así entonces, $F(\cdot)$ devolverá un resultado con el que tomar una decisión, para lo cual se suele recurrir a comparar ese resultado con un umbral (μ): si es superior a él se concede el crédito (D_1), y en caso contrario se deniegua (D_0), o viceversa. La forma matemática para expresarlo adoptaría la forma:

$$\vec{w}' \cdot \vec{x} \underset{<}{>} \mu \qquad (3.1.1)$$

La función $\vec{w}' \cdot \vec{x}$ cuando se compara con un umbral recibe el nombre de discriminante (que no deja de ser un tipo de clasificador).

La decisión mediante umbrales equivale a trazar una frontera de decisión en el espacio de observación que separa los clientes de ambas clases: a los que se les concedió y denegó el crédito.

Evidentemente, la decisión tomada puede no ser correcta y equivocarse con un cliente, pero lo que se persigue en este caso concreto no es minimizar el número de errores cometidos, si no optimizar un objetivo concreto, como es maximizar el beneficio. Puede que el banco prefiera equivocarse con varios clientes que solicitan un préstamo pequeño, que hacerlo con un cliente que solicita una cantidad de dinero mayor. Es decir, se busca que el decisor generalice lo mejor posible.

Persiguiendo una buena generalización, evitamos dos fenómenos muy dañinos en el entrenamiento máquina, representadas en la **Figura 3.1.3**: sobreajuste y subajuste.

El sobreajuste es el fenómeno que se produce en el entrenamiento de una máquina cuando nos ajustamos en exceso al conjunto de muestras de entrenamiento (etiquetadas) que tenemos, perdiendo así capacidad de generalización.

Y el subajuste es el fenómeno contrario al anterior, en este caso nos ajustamos demasiado poco a los datos de entrenamiento, y en consecuencia, extraemos menos información de la posible de los datos de entrenamiento.

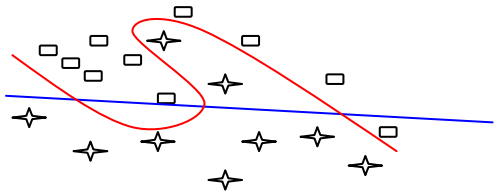

Figura 3.1.3 Población con muestras de dos clases separadas con una frontera roja que produce sobreajuste, y otra frontera azul que provoca subajuste.

Llegados a este punto en el que hemos descrito un caso concreto de clasificador lineal, el discriminante, detallamos los clasificadores lineales en cuanto a funcionamiento, usos y dos ejemplos sencillos, para acercarnos poco a poco a la solución final.

3.2. Clasificadores Lineales
Un ejemplo de clasificador lineal: el Perceptrón

3.2.1. Clasificadores Lineales

Antes de entrar de lleno en la explicación de lo que es un clasificador lineal, mencionamos un detalle fundamental de su funcionamiento, y es que el clasificador trabaja en cualquier tipo de dimensión (no consideramos la dimensionalidad del espacio) siempre y cuando los datos sean linealmente separables. Para el caso de que sean no separables, se aplicarían otras soluciones planteadas en próximos apartados.

Tal y como expusimos en la clasificación máquina, a partir de un conjunto de muestras de entrada $\vec{x}$ que tienen asociadas una etiqueta determinada y_i (salidas), el cometido del clasificador es asignar correctamente cada muestra a su clase, para delimitarlas mediante algún tipo de frontera, en este caso, lineal.

Matemáticamente podemos expresarlo así: definimos una función de valores reales de la forma $f : X \subseteq R^n \to R$ donde la clasificación de las muestras sigue la regla:

a partir de la entrada $\vec{x} = (x_1,...,x_n)'$ cada muestra se asigna según:

si $f(\vec{x}) \geq 0$ pertenece a la clase positiva

en caso contrario se asigna a la clase negativa

$f(\vec{x})$ podemos escribirla entonces:

$$f(\vec{x}) = \langle \vec{w} \cdot \vec{x} \rangle + b = \sum_{i=1}^{n} w_i \vec{x}_i + b \qquad (3.2.1)$$

siendo $(\vec{w}, b) \in R^n \times R$ los parámetros que controlan la función, y aplicando la regla de decisión dada por $\text{sgn}(f(\vec{x}))$, usando la convención de $\text{sgn}(0) = 1$.

La metodología de aprendizaje empleada para entrenar la máquina implica que los parámetros $\vec{w}$ y b se aprenden de los datos de entrenamiento, ya que al inicio del problema sólo disponemos de las entradas $\vec{x}$ y sus correspondientes salidas y_i. La ecuación de la frontera de decisión vendrá definida por $\vec{w}$ (vector normal al plano) y b (la ordenada), de manera que nuestros datos de entrada quedan divididos en dos partes por un hiperplano definido por la ecuación:

$$\langle \vec{w} \cdot \vec{x} \rangle + b = 0 \tag{3.2.2}$$

Comprobamos, efectivamente, cómo el hiperplano divide el espacio en dos mitades que corresponden a las muestras de dos clases distintas, tal y como apreciamos en la **Figura 3.2.1** donde una recta separa las cruces de los ceros.

El vector $\vec{w}$ se define perpendicular al hiperplano, y variando el valor del parámetro b desplazamos el hiperplano paralelo a sí mismo.

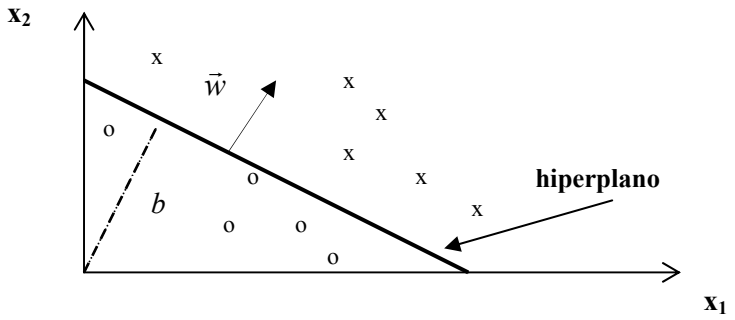

Figura 3.2.1 Conjunto de muestras de dos clases (ceros y cruces), donde se representan el hiperplano de separación entre ambas (línea continua en negrita), y los dos parámetros que lo definen: el vector $\vec{w}$ y la ordenada b.

Una vez presentado cómo funciona un clasificador lineal, incorporamos una nueva idea, la de la actualización *online* para describir uno de los algoritmos incrementales más importantes que mantiene la filosofía de los clasificadores lineales: el Perceptrón. Con él, también presentamos algunos conceptos interesantes que nos serán útiles más adelante.

3.2.2. Un ejemplo de clasificador lineal: el Perceptrón

El Perceptrón es uno de los algoritmos más empleados en la clasificación binaria que funciona como un discriminante o clasificador lineal. Posee las ventajas de que es muy fácil de programar, tiene baja carga computacional y, la más importante, actualiza su vector de pesos $\vec{w}$ cada vez que se encuentra con una muestra mal clasificada.

Sin entrar en detalles de su funcionamiento, la actualización de su algoritmo se produce mediante refuerzo Hebbiano (negativo), puesto que sólo corrige los errores. Si premiase los aciertos, sería positivo. En la **Figura 3.2.2** se muestra su representación en forma de diagrama de bloques [6].

Al ser un algoritmo muy sencillo, algunas de las limitaciones que presenta son:

- sólo converge si las muestras son linealmente separables, lo cual es lo mismo que decir que exista un hiperplano que clasifique correctamente las muestras de entrenamiento. En caso contrario, no existirá ningún hiperplano de separación y no convergerá;
- no ofrece buenas prestaciones de generalización aun siendo el problema linealmente separable, puesto que al corregir sólo los errores cometidos puede colocar la frontera en una posición poco adecuada;
- consigue un número reducido de errores sobre el conjunto de entrenamiento, lo que tampoco supone buena generalización;
- la convergencia del Perceptrón puede ser sumamente lenta, según un parámetro llamado margen que presentamos a contiuación, y, por lo tanto, que no nos sirva para nuestro propósito;

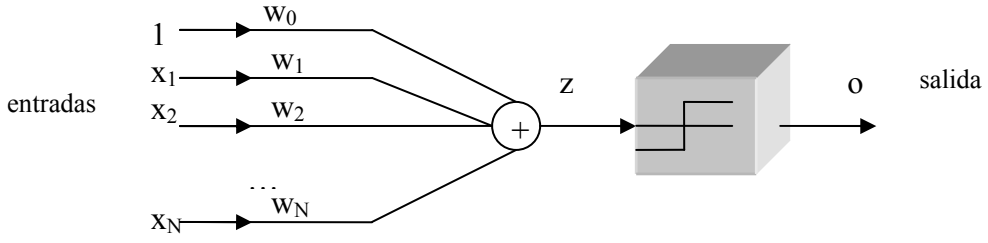

Figura 3.2.2 Perceptrón Monocapa “duro”

Aunque sus limitaciones hacen que no que podamos adoptarlo como solución final, ya comentamos que el Perceptrón resulta interesante para presentar algunos conceptos importantes, tales como margen o hiperplano de margen máximo, detallados en los siguientes párrafos.

En situaciones en que las muestras sean linealmente separables, el número de iteraciones que necesita para converger depende de un parámetro llamado margen (funcional), γ, el cual se define para una muestra $(\vec{x}_i, y_i)$ respecto a un hiperplano $(\vec{w}, b)$ como [7]:

$$\gamma_i = y_i\left(\langle \vec{w} \cdot \vec{x}_i \rangle + b\right) \tag{3.2.3}$$

de manera que si $\gamma_i > 0$, la muestra $(\vec{x}_i, y_i)$ está bien clasificada.

La distribución de margen (funcional) de un hiperplano $(\vec{w}, b)$ respecto a un conjunto de entrenamiento S es la distribución de márgenes de todo el conjunto de muestras de S, tal y como se aprecia en la **Figura 3.2.3**. En ocasiones, la distribución de margen mínimo se refiere al margen (funcional) de un hiperplano $(\vec{w}, b)$ respecto un conjunto de entrenamiento S [7].

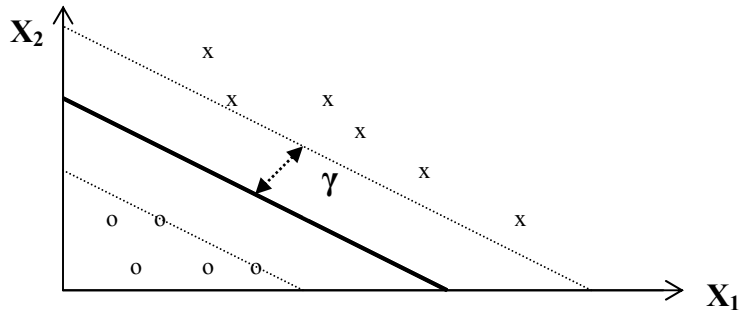

Figura 3.2.3 Representación del margen γ obtenido del conjunto de muestras x

En ambas definiciones, si reemplazamos el margen funcional por el margen geométrico obtenemos la cantidad equivalente para la función lineal normalizada $\left((1/\|\vec{w}\|)*\vec{w}, (1/\|\vec{w}\|)*b\right)$, que indica la distancia euclídea de los puntos desde el límite de

decisión en el espacio de entrada (siendo $\|\vec{w}\|$ la norma euclídea de $\vec{w}$). El margen geométrico será igual al margen funcional si el vector de pesos es el vector unidad [7]. En la **Figura 3.2.4** se muestra el margen geométrico para dos muestras.

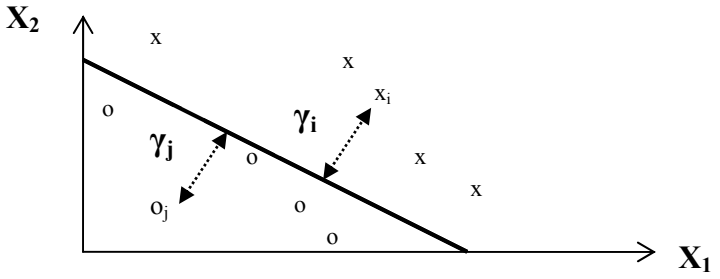

Figura 3.2.4 Representación del concepto de margen geométrico para dos muestras: γ_i para xi, γ_j para oj.

La última definición hace referencia al margen de un conjunto de entrenamiento S como el margen geométrico máximo sobre todos los hiperplanos. Y en concreto, el hiperplano que consigue dicho máximo se conoce como hiperplano de margen máximo. El tamaño de su margen será positivo para un conjunto de entrenamiento linealmente separable [7].

Para visualizar el concepto de hiperplano de margen máximo de forma gráfica nos apoyamos en la **Figura 3.2.5** donde apreciamos cómo las rectas discontinuas roja y azul separan correctamente ambos conjuntos de muestras y podrían ser perfectamente válidas como solución. Pero también apreciamos que no resultan del todo efectivas ya que están muy *pegadas* a uno de los dos conjuntos de muestras y hace que podamos obtener una mala generalización.

Con estas dos fronteras comprendemos mejor el problema del Perceptrón a la hora de establecer la frontera, pues aun siendo el problema linealmente separable, la coloca en una situación no óptima. Si adoptásemos como solución alguna de esas fronteras, correríamos el riesgo de que la siguiente muestra que analizásemos se encontrase antes o después de la frontera trazada y, por lo tanto, estaría mal clasificada.

Por el contrario, la recta continua negra más gruesa se posiciona en el punto medio entre las muestras más cercanas de clases distintas, convirtiéndose en la frontera que define el hiperplano de máximo margen y en la idónea para resolver el problema.

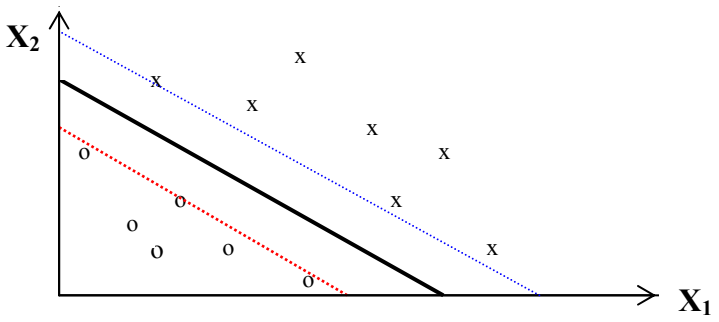

Figura 3.2.5 Distintos tipos de fronteras para un conjunto de muestras de dos clases, no "óptimas" en el caso de las rectas discontinuas de color, y frontera de margen máximo en negrita.

Evidentemente, ninguna de las fronteras del dibujo nos "salva" de que pueda aparecer otra muestra que nos obligue a moverla por encontrase mal clasificada, pero resulta fácil comprender que con el hiperplano de máximo margen corremos menos riesgo de que esto suceda, respecto a cualquiera de las otras dos fronteras discontinuas, gracias al hecho de situarse con la máxima separación posible entre ambas poblaciones.

Con lo visto hasta ahora, podemos quedarnos con algunas ideas interesantes como, la actualización del algoritmo sólo frente a errores, el empleo de la función cuantificadora para comprobar errores entre la etiqueta real de una muestra y la obtenida por el algoritmo, y las ventajas que parece aportar por ahora el hiperplano de máximo margen. Pero también tenemos que tener presentes las limitaciones que poseen los clasificadores lineales estudiados, como la no convergencia en escenarios no separables linealmente o su incapacidad de poder emplearse en situaciones *online,* para resolverlas más adelante.

Por ello, a continuación presentamos otro tipo de clasificador lineal ayudándonos de las máquinas de vectores soporte (SVM), los de máximo margen, para irnos acercando al diseño definitivo del filtro del igualador de canal.

3.3. Máquinas de Vectores Soporte (SVM) y Clasificador Lineal de Máximo Margen

3.3.1. Introducción a SVM

La tecnología de las máquinas de vectores soporte SVM (*Support Vector Machines*) fue elaborada por Vladimir Vapnik [31], cuando se trasladó en 1990 a Estados Unidos, y su equipo de los laboratorios AT&T [32].

El objetivo de las SVM surgió como respuesta a una necesidad común en el aprendizaje máquina, como es la clasificación de datos según un criterio concreto. La idea principal de las SVM consiste en tratar de separar un conjunto de muestras de *m* clases distintas con un hiperplano de dimensión *m*-1 [7]. De momento, nos centramos en el caso lineal.

En un principio las SVM se emplearon en la resolución de problemas de clasificación binaria (dos clases), pero poco a poco se ha ido comprobando su precisión y potencial en la resolución de cuestiones más complejas (problemas de clasificación multiclase o regresión) y actualmente las SVM, aparte de su aplicación en disciplinas de aprendizaje automático y reconocimiento de patrones, se usan en inteligencia artificial o minería de datos (*data mining)* [7].

Con lo dicho hasta ahora, podríamos definir a las SVM como un tipo de clasificador lineal, debido a que generan separadores lineales o hiperplanos en espacios de características muy complejos (si se aplican funciones de *kernel* complejas), con la máxima separación de margen posible.

Una característica muy importante de las SVM es la capacidad de minimizar el error de clasificación sobre datos nuevos, consiguiendo con ello buenos resultados de

generalización. Las SVM evitan así dos fenómenos ya presentados, como son el sobreajuste y el subajuste.

Dentro de los distintos tipos de SVM, el más simple es el clasificador de máximo margen que tratamos a continuación.

3.3.2. Clasificador Lineal de Máximo Margen

Empezamos mencionando que continuamos en situaciones en las que las muestras son linealmente separables, y recordando la definición de hiperplano de margen máximo respecto un conjunto de entrenamiento S como vimos en la **Figura 3.2.5**, donde el hiperplano de margen máximo es el que consigue mayor margen entre las muestras más próximas de clases distintas, mientras que las muestras más alejadas no resultan críticas en el diseño. O dicho de otra forma, es el hiperplano que maximice la distancia de separación entre los dos conjuntos de muestras, implicando el menor riesgo en la clasificación. Con ello, SVM busca el hiperplano de margen máximo en cada momento, y cuando surge un nuevo error, desecha la solución encontrada y calcula un nuevo hiperplano para adaptarse a la nueva situación.

Matemáticamente lo escribiríamos así:

$$\langle \vec{x}_i \cdot \vec{w} \rangle + b \geq 1 \text{ para } y_i = +1 \qquad (3.3.1)$$

$$\langle \vec{x}_i \cdot \vec{w} \rangle + b \leq 1 \text{ para } y_i = -1$$

La mínima separación que se consigue así entre los vectores y el hiperplano que fija la frontera es la unidad, y podemos resumir (3.3.1) en:

$$y_i \left(\langle \vec{x}_i \cdot \vec{w} \rangle + b \right) - 1 \geq 0, \text{ con } i = 1, \ldots, n \qquad (3.3.2)$$

A partir del clasificador descrito en (3.3.1), el margen geométrico γ resulta ser el margen funcional por medio de [7]:

$$\gamma = \frac{1}{2}\left(\left\langle \frac{\vec{w}}{\|\vec{w}\|_2} \cdot \vec{x}^{+} \right\rangle - \left\langle \frac{\vec{w}}{\|\vec{w}\|_2} \cdot \vec{x}^{-} \right\rangle \right) = \frac{1}{2\|\vec{w}\|_2} \left(\left\langle \vec{w} \cdot \vec{x}^{+} \right\rangle - \left\langle \vec{w} \cdot \vec{x}^{-} \right\rangle \right) = \frac{1}{\|\vec{w}\|_2} \qquad (3.3.3)$$

Descrito el clasificador que logra máximo margen y definido γ en (3.3.3), podemos presentar un aspecto fundamental en la construcción de las SVM: qué son y cómo se obtienen los vectores soporte con el siguiente ejemplo.

Dado un conjunto de muestras de entrenamiento linealmente separables $S = [(\vec{x}_1, y_1), ..., (\vec{x}_l, y_l)]$, el hiperplano $(\vec{w}, b)$ que resuelve el problema de optimización

$$\min_{w,b} \frac{1}{2}\left(\|\vec{w}\|\right)_2^2 \text{ [33]} \qquad (3.3.4)$$

$$\text{s.a. } y_i\left(\langle \vec{w} \cdot \vec{x}_i \rangle + b\right) \geq 1, \quad i = 1, \ldots, l$$

proporciona el hiperplano de margen máximo con margen geométrico $\gamma = \frac{1}{\|\vec{w}\|_2}$ [7]

Para resolver este problema de optimización, con las restricciones impuestas por (3.3.4), debemos utilizar multiplicadores de Lagrange $\alpha_i \geq 0,\ i = 1, \ldots, l$ [7] en su forma primaria, transformándolo en:

$$L(\vec{w}, b, \vec{\alpha}) = \frac{1}{2}\|\vec{w}\|^2 - \sum_{i=1}^{l} \alpha_i \left(y_i \left(\vec{w} \cdot \vec{x}_i + b \right) - 1 \right) \qquad (3.3.5)$$

Tomando derivadas parciales en (3.3.5) respecto $\vec{w}$ y b e igualando a cero llegamos a:

$$\vec{w} = \sum_{i=1}^{l} \alpha_i y_i \vec{x}_i \qquad \sum_{i=1}^{l} \alpha_i y_i = 0 \qquad (3.3.6)$$

Sustituyendo los resultados obtenidos de (3.3.6) en la forma primaria del Lagrangiano de (3.3.5), el problema de optimización se convierte en su equivalente dual como sigue:

$$\max W(\vec{\alpha}) = \sum_{i=1}^{l} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{l} \alpha_i \alpha_j y_i y_j \left\langle \vec{x}_i \vec{x}_j \right\rangle \qquad (3.3.7)$$

$$\text{s.a. } \sum_{i=1}^{l} y_i \alpha_i = 0, \quad \text{con } \alpha_i \geq 0\,,\ i = 1, \ldots, l$$

Entonces, el vector de pesos $\vec{w}^*$ que proporciona el hiperplano de margen máximo, con margen geométrico γ antes descrito, es:

$$\vec{w}^* = \sum_{i=1}^{l} y_i \vec{x}_i \alpha_i^* \tag{3.3.8}$$

Para hallar los valores de α^* hay que recurrir a las condiciones impuestas por el Teorema de Karush-Kuhn-Tucker [7] de forma que deben satisfacer:

$$\alpha_i^* \left[y_i \left[\left\langle \vec{w}^* \cdot \vec{x}_i \right\rangle + b^* \right] - 1 \right] = 0 , \quad i = 1, \ldots, l \tag{3.3.9}$$

Esto implica que sólo las entradas $\vec{x}_i$ para las cuales el margen funcional es uno, y por lo tanto se encuentran cercanas al hiperplano de separación, tienen su correspondiente α^* distinto de cero, y el resto de α^* serán nulos. De modo que en la expresión del vector de pesos $\vec{w}^*$ sólo se encuentran las entradas $\vec{x}_i$ con su consiguiente α^* no nulo, que son los llamados vectores soporte (*support vectors*).

Respecto a nuestro problema tenemos que hacer dos apreciaciones:

- el valor de los α_i^* asociados a las muestras que intervienen en la solución final del vector $\vec{w}^*$, serán calculadas según dos algoritmos distintos que se presentarán más adelante;
- la variable que controlará la restricción de margen máximo entre las muestras y el clasificador en nuestros experimentos se denotará β.

De modo que, los vectores soporte son las muestras críticas en el diseño del hiperplano de margen máximo, ya que son las que se encuentran más cercanas a dicho hiperplano. Lo podemos apreciar en la **Figura 3.3.1**, donde la recta más gruesa en negrita representa al hiperplano de margen máximo para un conjunto de muestras, con los vectores soporte resaltados en mayúsculas, mientras que el resto de muestras no afectan al diseño/situación de dicho hiperplano.

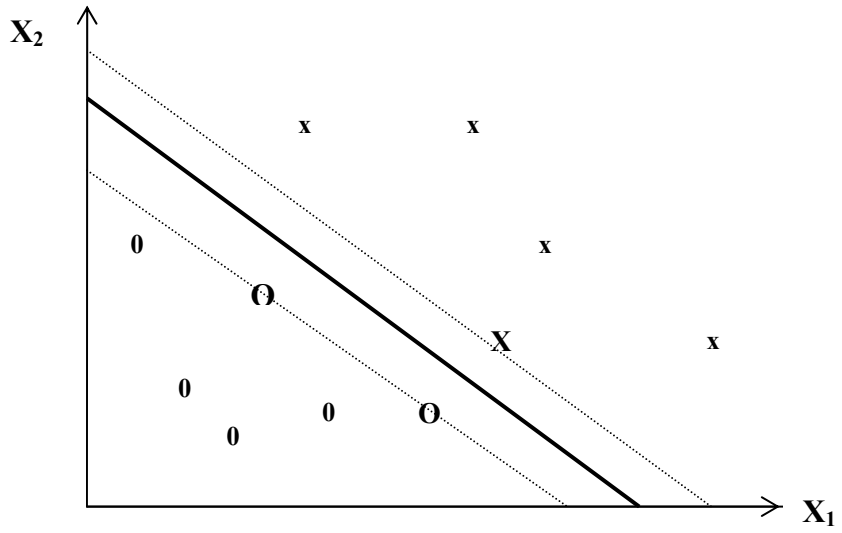

Figura 3.3.1 Ejemplo que muestra los vectores soporte para el siguiente clasificador de margen máximo

En el caso linealmente separable y con la figura que acabamos de presentar, vemos que los vectores soporte están bien clasificados puesto que siempre existirá un hiperplano que separe los conjuntos de muestras. Ahora veamos cómo cambia el problema en una situación donde las muestras no son separables linealmente.

3.4. Caso No Separable Linealmente

3.4.1. Introducción

Hasta ahora hemos tratado el caso de clasificadores lineales, que tienen la ventaja de ser muy sencillos en casos donde el problema es linealmente separable. Pero tienen la limitación del campo de acción para resolver problemas, puesto que la mayoría de éstos no pueden resolverse de manera lineal, como es nuestro caso.

Además, algunos clasificadores como el Perceptrón contaban con la característica de realizar un entrenamiento por épocas, o lo que es lo mismo, requerían analizar los datos varias veces (que serán muchas o pocas según su distribución) hasta llegar a la solución final, óptima o no como ya vimos. En situaciones *online* no puede realizarse un entrenamiento por épocas porque los datos varían continuamente y los análisis tienen que referirse a muestras actuales donde no existe la posibilidad de repetición. Es por esto, que en una situación de trabajo *online* la clasificación se convierte en adaptativa (variándola según los datos que se analicen) y no por épocas, donde las muestras pueden analizarse varias veces hasta encontrar la solución idónea.

También hay que contar con el hecho de que los datos contienen cierta componente de ruido, tal y como se vio en el capítulo anterior cuando describíamos las pérdidas y variaciones que sufre nuestra señal al ser transmitida. Esto puede provocar que si en un primer momento el problema era linealmente separable, ahora, a causa del ruido y dichas pérdidas, ya no lo sea y complique la solución.

Para resolver los problemas expuestos, primero tratamos la carencia de los clasificadores analizados respecto al caso en que los datos no sean separables linealmente, adoptando otra técnica que reemplaza la idea usada hasta ahora de maximizar el margen entre las muestras, por otra como el número de errores cometidos en la clasificación ayudándonos de las funciones *kernel*.

3.4.2. Clasificación mediante el número de errores

Para presentar el concepto del número de errores cometidos al clasificar un conjunto de muestras vamos a utilizar el siguiente ejemplo: en la **Figura 3.4.1** podemos ver que los datos no son linealmente separables debido a la presencia de una muestra, de la clase círculo de color violeta, que se encuentra muy alejada del comportamiento que mantienen el resto de muestras de su clase (a estas muestras se les llaman *outliers*). La sola presencia de esta muestra nos impide resolver el problema aplicando una frontera lineal, ya que cometeríamos como mínimo un error, obligándonos a emplear una

frontera cuadrática (línea roja discontinua). Sin otra alternativa, el clasificador lineal de máximo margen, empleado hasta ahora, no podría resolver este problema.

Para obtener una solución lineal tenemos que recurrir a una nueva metodología, tratando al *outlier* como lo que es, una muestra con comportamiento anómalo, para obviarla y aplicar una frontera lineal (recta de color azul). Con ello, estamos tolerando cierto margen de error en la clasificación a costa de emplear una frontera muy sencilla (que involucra menos coste computacional). Y vemos que la frontera lineal es la que garantiza el mínimo error posible en la clasificación del conjunto, pues no existe otra solución que nos proporcione una solución libre de errores empleando una forma lineal.

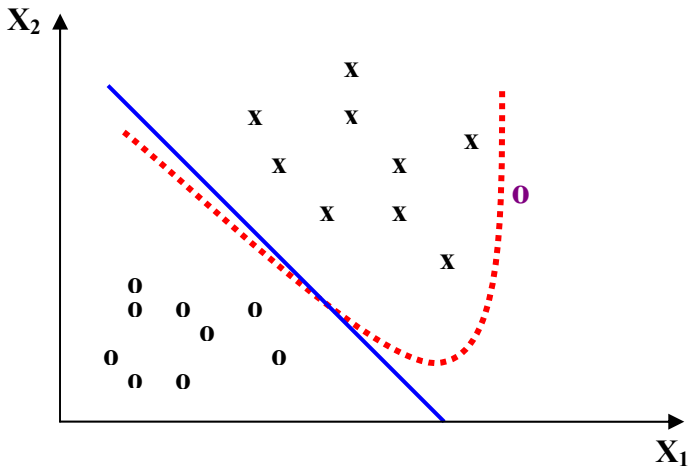

Figura 3.4.1 Ejemplo con la presencia de un *outlier* en el sistema, y dos soluciones posibles: la línea roja discontinua ofrece una solución libre de errores, y la línea azul contínua adopta la solución de mínimo error lineal posible

Con este ejemplo hemos demostrado las bondades de esta nueva forma de clasificación, y la incapacidad de los clasificadores lineales (de máximo margen o no) para enfrentarse a este tipo de problemas, puesto que se requiere, en este caso, una frontera cuadrática que dichos clasificadores no proporcionan. Un simple *outlier* en el problema impide emplear cualquier clasificador lineal.

Parece más conveniente, según el problema a resolver, admitir cierta tasa de error (tratando que sea baja), evitando clasificar correctamente determinadas muestras que no aportan (casi) nada a la solución final, que forzar su correcta clasificación

cuando puede que esto sólo nos lleve a empeorar la capacidad de generalización e incrementar el coste computacional. Con ello también evitamos emplear fronteras más complejas que no tienen la sencillez de las soluciones lineales, y nos impedirían seguir trabajando con clasificadores lineales.

Imaginemos qué nos reportaría clasificar bien el *outlier* de la **Figura 3.4.1**: una tasa de error nula nos hubiera obligado a emplear una formulación matemática más compleja para tratar una simple muestra, que puede darse de forma minoritaria. Parece buena práctica, en determinados casos, optar por el mal menor ignorándola y cometer cierto error.

Volviendo a la **Figura 3.4.1** vemos cómo cambia el concepto de clasificación entre los casos separables linealmente o no. Si en el primer caso se forzaba a situar la frontera de clasificación en la posición que maximizase el margen entre las muestras, en el segundo prima establecerla donde menos errores cometamos y luego, si se puede, maximizar el margen lo posible como se ha hecho en la figura.

El objetivo de minimizar el número de errores en la clasificación se desarrollará de manera más concreta cuando expongamos el *Tighter Budget Perceptron*, ya que existirán diversas técnicas de aplicar este principio.

3.5. Funciones de *Kernel*

Las funciones de *kernel* se usan para cubrir la carencia de los clasificadores lineales, en cuanto a su uso sólo en problemas linealmente separables, como un método para conseguir extensiones no lineales de algoritmos lineales. Por ello resultan una pieza fundamental de las SVM por su gran capacidad en la resolución de problemas complejos.

Por ejemplo, en problemas donde las muestras no sean linealmente separables, los clasificadores de máximo margen no consiguen separarlas sin error. Para corregir este déficit, un *kernel* suficientemente potente hace posible que cualquier problema sea separable, a costa de una función y coste computacional más complejos.

Veamos cómo se aplican.

3.5.1. Funcionamiento del *Kernel*

Una función *kernel* se verifica como tal si cumple el Teorema de Mercer [7].

Al aplicar una función *kernel*, idealmente se trata de elegir una función que transforme el espacio que contiene los datos de entrada en un nuevo espacio que facilite su resolución para delimitar las muestras de las distintas clases. De modo, que el *kernel* induce una transformación de la forma:

$$\vec{x} = (x_1,...,x_n) \rightarrow \vec{\phi}(\vec{x}) = (\phi_1(\vec{x}),...,\phi_n(\vec{x})) \qquad (3.5.1)$$

Este paso consiste en proyectar un espacio de entrada X en un nuevo espacio transformado $F = \{\phi(\vec{x}) \mid \vec{x} \in X\}$ [7].

En la **Figura 3.5.1** se muestra la proyección de un espacio de entrada de dos dimensiones en un nuevo espacio de dos dimensiones: en el espacio de entrada $\vec{X}$ las muestras no pueden separarse por una función lineal, y aplicando la transformación ϕ, conseguimos que en el nuevo espacio transformado $\vec{F}$ sí sea posible.

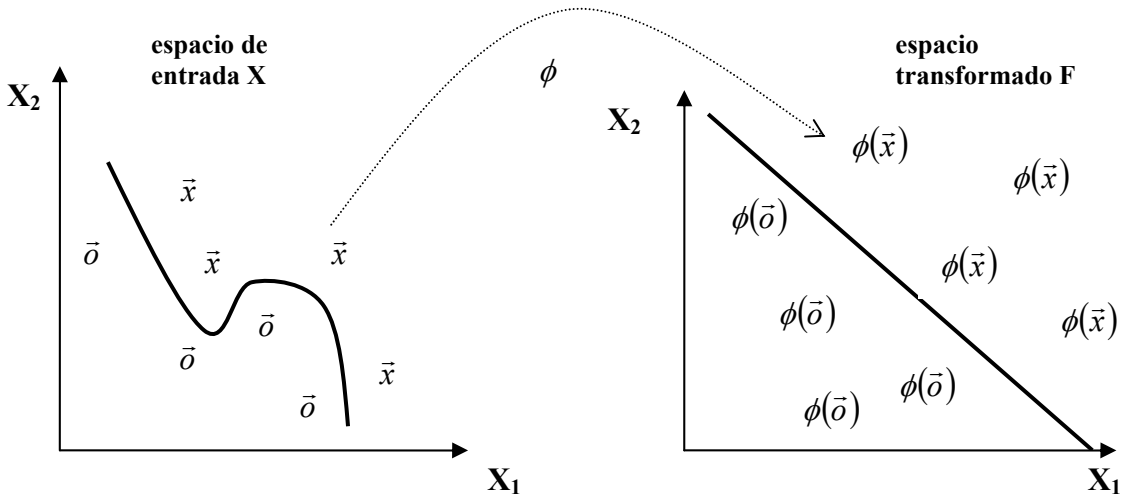

Figura 3.5.1 Espacios original y transformado mediante una función *kernel* para aplicar frontera lineal

El uso de *kernels* lleva a que construyamos una función de la forma:

$$f(\vec{x}) = \sum_{i=sv} \alpha_i K(\vec{x}_i, \vec{x}) \qquad (3.5.2)$$

donde los elementos del *kernel* son el vector soporte correspondiente y el conjunto de símbolos. Posteriormente se mejora esta ecuación para no considerar todos los símbolos, sino un conjunto concreto según distintos criterios que establezcamos para mejorar la solución final, como utilizar sólo un conjunto de tamaño *T*.

3.5.2. *Kernel* empleado

El tipo de *kernel* que usamos en los experimentos responde a la forma:

- Gaussiano: contenido dentro de las funciones RBF (*Radial Basis Function)* [7]:

$$K(\vec{x}, \vec{x}') = \exp\left(-\frac{\left\| \vec{x} - \vec{x}' \right\|^2}{2\eta^2} \right) \qquad (3.5.3)$$

siendo η un parámetro

3.5.3. Aplicación de la función *Kernel* en nuestro problema

El empleo de las funciones de *kernels* en nuestro algoritmo se aplica cuando tenemos que rehacer los cálculos de *predicción* de futuros símbolos en dos situaciones

que obligan a reajustar la máquina (algoritmo), tal y como explicamos en la introducción: si se viola la restricción impuesta por β o cometemos un error entre el símbolo real y el obtenido por nosotros.

La forma de operar con los *kernels* lo mostramos con un ejemplo: supongamos que nuestro problema tiene 100 símbolos no separables linealmente y sabemos que los vectores soporte ideales son los símbolos [15 40 75], que almacenamos en un array. Es decir, estos tres símbolos nos garantizan clasificar el conjunto total con la menor tasa de error posible.

Si además del instante actual consideramos los tres anteriores para el manejo de los vectores soporte a la hora de realizar los cálculos, quedarían como sigue:

vector soporte 1, vs1: [15 14 13 12]

vector soporte 2, vs2: [40 39 38 37]

vector soporte 3, vs3: [75 74 73 72]

Ahora, tenemos que construir el vector de *predicciones* para todos los símbolos con dichos vectores soporte con la siguiente estructura: definimos una matriz, X, de tamaño 100x3 (100 muestras, 3 vectores soporte), donde cada elemento de la matriz se calcula con la función *kernel* ya presentada. La función que construimos tiene la forma tal y como describimos en la ecuación 3.5.2 empleando el *kernel* descrito en 3.5.3, de manera que cada elemento de la matriz X se calcula como sigue:

Los elementos de la matriz se construyen como sigue:

$X(1,1) = K\,(\,X(1,:),\,vs1\,) = \exp\,[-\,((1-15)^2 + (0-14)^2 + (0-13)^2 + (0-12)^2)\,/\,2\eta^2\,]$

$X(1,2) = K\,(\,X(1,:),\,vs2\,) = \exp\,[-\,((2-15)^2 + (1-14)^2 + (0-13)^2 + (0-12)^2)\,/\,2\eta^2\,]$

$X(1,3) = K\,(\,X(1,:),\,vs3\,) = \exp\,[-\,((3-15)^2 + (2-14)^2 + (1-13)^2 + (0-12)^2)\,/\,2\eta^2\,]$

$X(2,1) = K\,(\,X(2,:),\,vs1\,) = \exp\,[-\,((1-40)^2 + (0-39)^2 + (0-38)^2 + (0-37)^2)\,/\,2\eta^2\,]$

$X(2,2) = K\,(\,X(2,:),\,vs2\,) = \exp\,[-\,((1-40)^2 + (0-39)^2 + (0-38)^2 + (0-37)^2)\,/\,2\eta^2\,]$

…

$X(100,3) = K\,(\,X(100,:),\,vs3\,) = \exp\,[-\,((100-75)^2 + (99-74)^2 + (98-73)^2 + (97-72)^2)\,/\,2\eta^2\,]$

el proceso se repite hasta completar la matriz completa para los 100 símbolos y tres vectores soporte. De esta forma, la primera columna de la matriz se refiere a la *predicción* para cada símbolo hecha con el primer vector soporte; la segunda columna

es la *predicción* de cada símbolo a partir del segundo vector soporte; y la tercera columna la *predicción* operando con el tercer vector soporte. A su vez, cada columna va multiplicada por la constante, α, relacionada con la etiqueta del vector soporte correspondiente. Con ello creamos el vector de predicciones, $\vec{d}$: $X \cdot \vec{\alpha} = \vec{d}$.

Supuesto para este caso que conocemos las etiquetas (sus signos) correctas de todos los símbolos almacenados en $\vec{y}$, sólo queda calcular el vector de errores: $\vec{e} = \vec{y} - \vec{d}$. Las posiciones de $\vec{e}$ distintas de cero, son los errores cometidos.

Hemos detallado el caso en que conocemos a priori los símbolos que son vectores soporte (que implican menor error), lo cual no sucede en nuestro caso, pues en un problema *online* no se puede saber qué símbolos son vectores soporte, sino que hay que ir ajustando la *predicción* cada vez que aparece un nuevo vector soporte. Buscamos, pues, los mejores vectores soporte posibles en cada momento, lo que nos obliga a ir actualizando el array de los vectores soporte para guardar los mejores y aplicar alguna técnica para saber cuáles son los mejores en cada momento.

Otra diferencia entre el ejemplo estudiado y nuestro problema es que empleábamos todos los vectores soporte para calcular todos los elementos de la matriz, mientras que en nuestro caso carece de sentido hacerlo, ya que la *predicción* se realiza para símbolos futuros, veamos cómo cambia respecto el ejemplo tratado: si tenemos los vectores soporte [15 40] y al analizar el símbolo 75 también lo es, lo incluiremos en el array para calcular la *predicción* de futuras muestras desde la 76 hasta la 100, pues los símbolos pasados ya se han analizado previamente. Con ello, los cálculos de *predicción* para los símbolos 76 hasta el 100 serán el resultado de sumar los productos que se obtengan individualmente entre cada vector soporte, [15 40 75], y su correspondiente α con los símbolos 76 hasta el 100.

Uno de los problemas que tenemos que manejar es la alta carga computacional que se puede alcanzar, si la caché es suficientemente grande y existen numerosos símbolos que sean vectores soporte. Esta y otras cuestiones las resolveremos con el algoritmo *Tighter Budget Perceptron*.

3.6. Solución propuesta: *Tighter Budget Perceptron*

3.6.1 Introducción

Llegados a este punto, es hora de abordar la solución final para resolver el problema expuesto. A lo largo de todo este capítulo hemos ido desarrollando la teoría de distintos clasificadores lineales en situaciones donde los símbolos eran o no linealmente separables. Pero todavía no hemos hablado de ningún algoritmo que se emplee en problemas de clasificación *online*, tal y como nos interesa, ya que los clasificadores lineales analizados no están preparados para tal situación al necesitar todo el conjunto de muestras para trabajar.

Otro punto que de momento no tiene respuesta es el almacenamiento de los vectores soporte. En el punto anterior mencionamos que se deben limitar de algún modo, para impedir guardar una cantidad excesiva que consumiese gran cantidad de memoria. Ahora veremos dos técnicas para controlar su número.

Primero hablaremos del algoritmo desarrollado en el artículo *Online Classification on a Budget* [34] que muestra una opción para saber qué muestras se incluyen en la caché y cuáles se extraen cuando se llene, y a continuación del algoritmo *Tighter Budget Perceptron*, surgido como mejora del anterior, ofreciendo dos versiones para saber qué vectores soporte se sacan de la caché, que se encargará de almacenar los vectores soporte y será definida a continuación en el siguiente apartado.

3.6.2 Antecedentes: *Online Classification on a Budget*

Para resolver el problema de la alta carga computacional de las SVM, surgieron algoritmos alternativos que reducen el número de vectores soporte. La pega que le

encontramos es que, aunque se trate de un algoritmo en bloque, se accede a cada muestra de entrenamiento varias veces, iterando sobre cada una a lo largo de varias pasadas. De manera que no se ajusta a nuestro propósito de aplicación en situaciones *online*.

El algoritmo de este artículo tomó como modelo al Perceptrón pero modificando su funcionamiento: si el Perceptrón obtenía $\vec{w}$ como una combinación lineal de los vectores soporte sin establecer un tamaño máximo, el algoritmo propone fijarlo a un tamaño (N) que podemos usar para los cálculos de las *predicciones*.

Ahora el problema está en administrar ese tamaño N, y es donde interviene el margen. Sin meternos en un repaso profundo de las ecuaciones con las que trabaja este algoritmo, éste emplea dos procesos, inserción y borrado de pesos para controlar N: si una muestra es errónea la añade en la caché (fase de inserción), y si la caché está completa, la revisa buscando el vector soporte que tenga mayor margen para borrarla (fase de borrado).

De esta forma, se mejora la generalización puesto que la solución depende de un menor número de muestras (las que tengan menor margen) respecto a si no limitásemos el tamaño de la caché. Además reducimos considerablemente la carga computacional respecto la memoria consumida, pues es fácil entender que a menor tamaño de la caché, menos operaciones se requieren, menos cálculos se aplican y menos memoria se consume.

Ahora bien, el motivo por el cual no adoptamos esta solución como válida reside en el hecho de que en nuestro problema carecemos del parámetro margen para saber qué símbolos “eliminamos” de la caché al encontrarnos en una situación no separable linealmente. De ahí que nos veamos en la necesidad de adoptar otra técnica para nuestro problema.

3.6.3 *Tighter Budget Perceptron*

Puesto que los vectores soporte son la parte fundamental para resolver el problema, este algoritmo se ideó con la intención de almacenar los más representativos para una correcta clasificación.

Por ello, se mantiene la filosofía de limitar el número máximo de la caché (*N*) y eliminar según un criterio concreto cuando se supere el tamaño máximo fijado.

Para saber qué muestra sacamos de la caché se realiza la siguiente comprobación: para cada uno de los vectores se comprueba la cantidad de errores que cometería el clasificador resultante si eliminamos dicho vector soporte de la solución final. Una vez conocidos los errores de cada vector soporte, sacaríamos la muestra que provoque menor error puesto que resulta ser la menos crítica en el diseño final [1].

De forma matemática quedaría como sigue:

$$i = \arg\_\min_{j \in c_t} \left\{ \sum_{k=1}^{t} L\left(y_k, sign\left(\left(\vec{w}_{t-1} - \alpha_j y_j \vec{x}_j\right)\vec{x}_k\right)\right)\right\} \qquad (3.6.1)$$

Con la siguiente **Figura 3.6.1** intentamos explicar más claramente de forma gráfica qué vector soporte se elimina según la ecuación (3.6.1) si la caché está completa en un problema *online*.

Definimos una caché inicial de tamaño cuatro, de forma que tras examinar toda la población de muestras se establece que los vectores soporte son las muestras de color azul numeradas [**X_1, 0_2, 0_3, X_4**], y la frontera óptima lineal en la situación inicial está marcado con línea negrita más gruesa. Esta frontera provoca una tasa de error de 1/35 (**X_4**).

Manteniendo el principio de que debemos modificar la máquina cuando se comete un error (el signo de nuestra *predicción* y el real no coinciden) o el valor de *predicción* del símbolo actual es menor que β, si la siguiente muestra a clasificar es la cruz de color rojo, **x**, sucede que: debemos desplazar la frontera inicial para clasificarla bien, y si además suponemos que la distancia entre dicha muestra y el clasificador inicial es menor que nuestro parámetro β, resulta ser candidata a convertirse en vector

soporte. Como la caché está completa, hay que calcular qué muestra suprimimos de la caché.

Para ello, el algoritmo simula la "eliminación" de cada uno de los vectores soporte [$\mathbf{X_1}$, $\mathbf{0_2}$, $\mathbf{0_3}$, $\mathbf{X_4}$, **x**] de forma individual, para calcular según (3.6.1) cuántos errores se cometerían al reclasificar, sin considerar dicha muestra, el resto de muestras. Las posibles fronteras darían como resultado cometer los siguientes errores: para $\mathbf{X_1}$ 2 errores ($\mathbf{X_4}$ y **x** ó **x** y $\mathbf{0_3}$), para $\mathbf{0_2}$ 1 error ($\mathbf{X_4}$), para $\mathbf{0_3}$ 1 error (**x**), para $\mathbf{X_4}$ 1 error ($\mathbf{0_2}$ ó **x**), para **x** 1 error ($\mathbf{X_4}$).

Vemos que, sin el vector soporte $\mathbf{X_1}$, cometeríamos más errores al reclasificar el conjunto de muestras, lo que indica que es una muestra crítica que no debe sacarse de la caché. Mientras que para el resto de vectores soporte, como todos tienen un único error, podemos descartar cualquiera de ellas. Al no haber un criterio establecido en caso de igualdad de errores cometidos entre los vectores soporte, se opta por descartar los más antiguos, en este caso $\mathbf{0_2}$, y la caché de vectores soporte quedaría con [$\mathbf{X_1}$, $\mathbf{0_3}$, $\mathbf{X_4}$, **x**].

La nueva frontera de color rojo discontinuo queda establecida una vez conocida la nueva caché de vectores soporte. Eso sí, la nueva frontera tiene una tasa de error de 2/36 ($\mathbf{X_4}$, **x**).

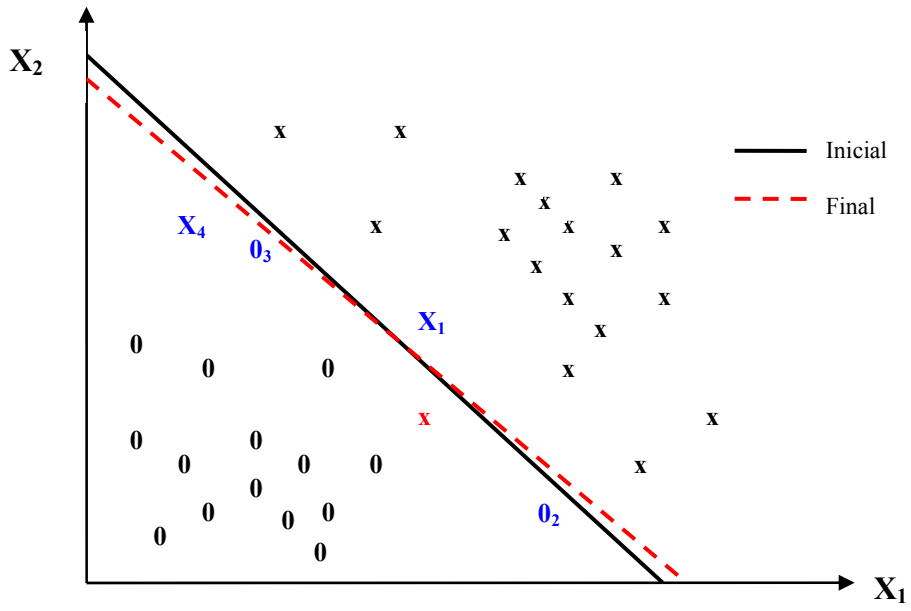

Figura 3.6.1 Ejemplo de simulación de la caché completa empleando *Tighter Budget Perceptron*

Ahora tenemos que tratar un nuevo aspecto del diseño que consiste en definir sobre qué conjunto de muestras trabajamos para decidir qué vector soporte sale de la caché, lo que nos abre nuevas variantes de diseño.

Tighter Budget Perceptron propone una caché secundaria que guarde las muestras sobre las que avaluar los vectores soporte para saber cuál se extrae. La caché secundaria puede contener:

- todas las muestras, tal y como hemos hecho en el ejemplo anterior;
- las vistas hasta el instante actual más cercanas al margen, es decir, sólo los vectores soporte.

La primera opción desemboca en un incremento de la carga computacional como consecuencia de operar sobre todos los símbolos, sobre todo si la población total es muy grande. Aunque cabe destacar, que dicha regla sólo se aplica cuando el símbolo analizado tiene un margen menor que β (indica que es candidato a ser vector soporte), lo que implica que su aplicación en problemas donde los símbolos estén poco afectados por el ruido conlleve pocas operaciones porque la mayoría se concentrará en torno a las muestras de su población.

En contrapartida, el incremento de la carga computacional se ve compensada por el hecho de que al manejar todas y almacenar sólo un conjunto de las muestras críticas, el clasificador resultante tiene mejores propiedades de generalización.

La pega es que en nuestro caso no podemos garantizar que nuestros símbolos estén poco afectados por el ruido, todo lo contrario, ya estudiamos que el efecto Doppler y el ruido AWGN comportarán grandes cambios en la información manejada. Además al tener que trabajar con $115.2 \cdot 10^3$ símbolos puede suponer un problema en el manejo de las matrices a la hora de los cálculos a realizar. De modo que aunque construir un clasificador generalista siempre es positivo, las características de nuestro medio y la cantidad de datos a manejar suponen dos serios problemas.

En cuanto al segundo método propuesto y manejar sólo los vectores soporte, se reduce considerablemente el coste computacional. Sin más que aplicar un pequeño cambio a la ecuación (3.6.1) se obtiene [1]:

$$i = \arg\_\min_{j \in c_t} \left\{ \sum_{k \in c_t} L\left(y_k, sign\left(\left(\vec{w}_{t-1} - \alpha_j y_j \vec{x}_j\right)\vec{x}_k\right)\right)\right\} \qquad (3.6.2)$$

La desventaja que se le puede encontrar a este modelo, es que al tratar sólo con los símbolos de la caché puede resultar no del todo óptimo, puesto que estamos obviando el resto.

De modo, que buscando el equilibrio entre el posible aumento de carga computacional del modelo (3.6.1) y la alternativa poco generalista de (3.6.2), nosotros planteamos una variante para saber qué vector soporte "eliminar" de la caché: redefinimos la caché secundaria como una ventana con *T* símbolos anteriores al actual, haciendo que se convierta en una ventana deslizante que se vaya desplazando a medida que se va resolviendo el problema (de manera adaptativa), y "eliminar" el vector soporte de dos formas posibles.

La primera propone que sea el que nos deje un clasificador de mínimo error en las muestras definidas en nuestra ventana de tamaño *T*. La cuestión será encontrar ese tamaño *T* idóneo que desemboque en el mínimo error posible, para lo que deberemos realizar varias pruebas para encontrarlo y hallar la importancia del pasado reciente para predecir el valor de futuros símbolos en este tipo de problemas *online*.

La otra variante consiste en olvidarse de la ventana de tamaño *T* y cuando se tenga que sacar un vector soporte de la caché, se elige el más antiguo suponiendo que las muestras más antiguas apenas influyen sobre las actuales. Se sigue el mismo principio que con la ventana deslizante, pero "borrando" directamente el más antiguo sin realizar cálculos.

Con los dos métodos propuestos, incorporamos un nuevo parámetro para resolver el problema y averiguar cuál conduce al mínimo error.

Resuelto el cómo almacenar los vectores soporte, ahora enfocamos nuestra atención en el último parámetro a controlar, α_t, que ayuda en la *predicción* del signo de cada muestra. Su valor se actualiza de acuerdo a dos algoritmos distintos que probaremos para averiguar cuál se adapta mejor al problema, como son:

- el Perceptrón clásico, que asigna $\alpha_t = 1$;
- y el SVM sin sesgo, que establece $\alpha_t = \min\left(C, \max\left(0, \frac{1 - y_i\left(\vec{w} \cdot \vec{x}_i\right)}{\vec{x}_i \cdot \vec{x}_i}\right)\right)$

  con β=1 (margen del clasificador), y *C* un parámetro de control que impide que α pueda alcanzar un valor alto.

Capítulo 4

Experimentos

Después de relatar en los capítulos 2 y 3 toda la parte teórica, ahora nos centramos en la aplicación práctica mediante la descripción de los experimentos llevados a cabo, detallados en el primer punto de este apartado.

A continuación vendrán los resultados numéricos y gráficos obtenidos para comparar los experimentos realizados con distintos parámetros y la discusión de los resultados.

4.1. Descripción de los experimentos

Los experimentos se realizarán en el entorno de desarrollo Matlab para estudiar los dos algoritmos de actualización de la caché de vectores soporte planteados en el apartado anterior: saber qué vector soporte de la caché principal se elimina cuando esté completa mediante una ventana de tamaño *T*, u optar por eliminar directamente el más antiguo.

Los tres principales parámetros que manejamos son:

- los tres canales propuestos;
- las dos técnicas mencionadas en el apartado 3.6.3 que modifican el valor de α, Perceptrón y SVM sin sesgo;

- el valor de la relación SNR, para así comprobar cómo varía el error medio de cada prueba en función de dicho valor.

El resto de parámetros que manipulamos para terminar de afinar los resultados son:

- la constante β, para controlar el margen;
- la constante η de la función RBF;
- el tamaño de la caché (*N*) principal de vectores soporte que guarda las muestras críticas en el diseño;
- el número de símbolos previos de cada vector soporte utilizados para los cálculos de las predicciones mediante la función RBF;
- el tamaño de la ventana (*T*), en caso de optar por la resolución con este método.

Las pruebas con menor BER indican que los valores empleados en los parámetros son los indicados para dicha situación.

Presentados los parámetros de configuración de las pruebas, pasamos a mostrar los resultados obtenidos.

4.2. Resultados

Dividiremos los resultados por apartados en función de los distintos canales estudiados, para, dentro de cada canal, diferenciar según las técnicas empleadas (Perceptrón o SVM sin sesgo) y por último si se ha optado por utilizar el programa que elimina el vector soporte más antiguo o emplea una ventana de tamaño *T*.

Primero detallamos los experimentos que resuelven el problema mediante la ventana de tamaño *T* para saber qué vector soporte sale de la caché principal (apartados 4.2.1 → 4.2.6), y seguiremos con la técnica de eliminación (apartados 4.2.7 → 4.2.12).

En todos estos apartados, las pruebas realizadas emplearán el seguimiento por pilotos, es decir, en todo momento transmisor y receptor tendrán constancia del símbolo que se ha transmitido. Esta situación es más propia de un problema de *machine learning*, pero con ello averiguaremos el funcionamiento de los algoritmos en una situación ideal, ya que en la vida real nunca sucede que transmisor y receptor tengan conocimiento de la información que se transmite puesto que no tendría sentido realizar la comunicación.

Cuando conozcamos cómo se defienden nuestros algoritmos en este escenario teórico y conozcamos los parámetros de mínimo error, plantearemos en el apartado 4.3 una situación propia de un problema de comunicaciones, donde sólo tendremos acceso a una pequeña cantidad de símbolos, llamados pilotos, mientras que el resto serán desconocidos para el receptor. Para resolver el problema en este caso, en la parte de pilotos el algoritmo se guiará por el símbolo que ve en ese momento accediendo directamente a su valor para realizar los cálculos de igualación, y en el resto aplicará la técnica de guiado por decisión, tal y como se explicó en el Capítulo 1 de esta memoria.

Como nota para todas las tablas de resultados, mencionamos que los destacados en negrita se refieren a aquellos resultados sobre los que realizaremos comentarios, comparaciones o extraeremos conclusiones, los azul oscuro representan los experimentos adoptados como solución final en cada caso, y los verde turquesa los de mínimo error alcanzado pero rechazados como solución.

4.2.1. Canal 1 Perceptrón con ventana

Comenzamos mostrando las tasas de error obtenidas para el valor más alto de SNR estudiado (50 dB) mediante la técnica de Perceptrón, empleando la solución que define una ventana de tamaño T y saber sobre qué muestras trabajar para eliminar el vector soporte apropiado de la caché principal.

Nº experimento	Tamaño Caché	η (Kernel)	β	Nº muestras empleadas en predicción	Tam. Ventana T	Tasa Error (%)
1	75	1	0,1	4	500	4,4179
2	**150**	**1**	**0,1**	**4**	**500**	**4,0082**
3	**150**	**0,5**	**0,1**	**4**	**500**	**3,8128**
4	**150**	**1**	**0,1**	**3**	**500**	**6,4242**
5	**150**	**1**	**0,001**	**6**	**500**	**3,7955**
6	**150**	**1**	**0,1**	**6**	**500**	**3,147**
7	**150**	**1**	**0,01**	**6**	**500**	**3,4803**
8	**150**	**1**	**0,1**	**6**	**100**	**3,305**
9	**150**	**1**	**0,1**	**6**	**750**	**3,3605**
10	**150**	**1**	**0,1**	**6**	**300**	**3,1331**
11	**150**	**1**	**0,1**	**6**	**250**	**3,1973**
12	**150**	**1**	**0,1**	**6**	**1000**	**3,4352**
13	**250**	**1**	**0,1**	**6**	**300**	**2,7589**
14	150	1	0,1	6	350	3,1982
15	**150**	**1**	**0,1**	**5**	**500**	**2,6721**
16	150	1	0,1	5	300	2,7363
17	150	1	0,01	5	200	3,1366
18	200	1	0,1	5	300	2,7355
19	250	1	0,1	5	300	2,6452
20	**250**	**1**	**0,1**	**5**	**300**	**2,5896**
21	**250**	**1**	**0,01**	**5**	**300**	**3,1218**
22	**250**	**1**	**1**	**5**	**300**	**2,0523**
23	**250**	**1**	**2**	**5**	**300**	**2,0566**
24	250	0,5	0,1	5	300	2,4334
25	250	0,2	1	5	300	2,3648
26	**250**	**1**	**0,1**	**7**	**300**	**3,4786**

Tabla 1: Resultados Canal 1 Perceptrón con ventana 50 dB

Comencemos explicando porqué hemos adoptado el experimento 20 como solución en lugar del 22. La única diferencia entre ambos es el valor de β (0,1 ó 1) y el motivo por el que no adoptamos como solución al 22 se debe a la cantidad de tiempo requerida por dicha prueba para resolver el problema, que hemos medido con el contador de tiempo del propio software de desarrollo. Se aprecia que la disminución de error en términos absolutos entre ambos es mínima, pero en cambio el 22 necesita en torno a las seis horas para finalizar frente a las dos horas del número 20. Es por ello, que no nos parece oportuno tomar el valor de $\beta = 1$ como valor ideal, si valiendo 0,1 el error es casi el mismo y ganamos tiempo y ahorramos energía.

La explicación matemática para justificar la cantidad de tiempo necesario para hacer la prueba se debe a que β actúa a modo de discriminante para saber si el símbolo tratado es vector soporte o no. Si se escoge un valor de β alto en comparación con los valores de los símbolos, implica realizar gran cantidad de operaciones con la caché que consumen muchos recursos y tienen su coste en un aumento del tiempo computacional. Lo que implica que se debe coger un discriminante acorde al valor de los símbolos, y que provoque una tasa de error asumible.

El problema añadido que tendría tomar el valor $\beta = 1$, es que si en el mejor caso posible de SNR el tiempo ya es excesivo, a medida que la SNR disminuya el tiempo se incrementará notablemente dado que los valores están más corruptos, y ello supondría una pérdida de tiempo, energía y dinero importante para el resto de pruebas a realizar. Lo cual no nos parece oportuno.

Con la prueba 23, sucede lo mismo que con la 22, el tiempo requerido es sumamente alto y, por ello, se descarta como solución.

En cuanto al resto de experimentos, se aprecia cómo hemos ido perfeccionando el error obtenido, empezamos en una tasa del 4,4% y disminuimos hasta un 2,58%, es decir, que encontrando los parámetros óptimos se ha conseguido bajar el error un 41,38%, que no resulta despreciable.

Los resultados nos dicen que con cada vector soporte encontrado el número ideal de símbolos empleados para la predicción en este caso es 5, y que utilizando más o menos no estamos consiguiendo nueva información para predecir el canal y sólo provocamos un aumento del error final. El mejor resultado que se consigue con 6

símbolos es del 2,7589%, que aunque muy próximo al 2,5896% elegido, es ligeramente superior y se traduce en que cualquier otra combinación de valores en el resto de parámetros, el error sube por encima del 3% tal y como puede verse en pruebas 8…12.

Lo que nos indica además el resultado obtenido por los experimentos 13 y 26, es que no nos aporta información útil el emplear más símbolos para la igualación de canal, dado que la tasa de error aumenta en ambos casos para las mismas condiciones que el experimento 20. Con las pruebas 2, 4, 6, 15 y 26, que comparten los mismos valores salvo el número de muestras utilizadas para la predicción, se aprecia claramente que cinco símbolos son los adecuados para los cálculos sin subajustar o sobreajustar el modelo.

Por lo tanto, manejando 5 símbolos encontramos una solución equilibrada entre el tiempo empleado y el error, eliminamos redundancia de la información que recibimos por causa del multitrayecto y conseguimos recuperar la señal de manera más eficaz.

El tamaño de la ventana más adecuado ha resultado de 300, lo que significa que verificando el error cometido en esa pequeña fracción de símbolos anteriores obtenemos menor error.

Analizando los resultados y verificando como hemos llegado a estas conclusiones sobre cómo ir manipulando los distintos parámetros, podemos apreciar que cambiando sólo el tamaño de la caché, el error disminuye a medida que incrementamos el tamaño. Esto nos indica que si utilizamos un tamaño lo suficientemente grande, la caché nos ayuda a igualar mejor gracias a que tenemos un conjunto amplio sobre el que averiguar qué muestras son más importantes. Con ello, al final se obtiene que con 250 vectores soporte la tasa de error es la más baja posible.

El hecho de no poder usar cachés mayores se debe a que la cantidad de memoria requerida por el ordenador para realizar los cálculos y saber qué muestra eliminar de la caché con este algoritmo empleado, provoca que se desborde la memoria en dichas pruebas al requerir cálculos muy complejos computacionalmente para encontrar sobre la ventana definida qué vector soporte conlleva menor error. Por ello, se descarta usar cachés superiores a 250.

El valor de ventana óptimo para resolver el problema se ha fijado en 300 después de realizar varias pruebas modificando sólo este parámetro y manteniendo los mismos valores en el resto (experimentos 8, 9, 10, 11, y 12). A continuación, una vez descubierto que debemos utilizar cinco símbolos para los cálculos para la igualación,

verificamos de nuevo que efectivamente el tamaño óptimo de la ventana es de 300 tras descartar otros valores.

Lo que nos indica la ventana es que dicho conjunto de muestras influyen más para resolver el problema, y que si nos fijamos en otros más amplios (que incluye símbolos muy lejanos) o más pequeños (símbolos demasiado recientes respecto el instante actual), sólo empeoramos la solución. La explicación consiste en que dada la situación en que nos encontramos (canales móviles, múltiples reflexiones y velocidad elevada) lo más normal es que el instante actual no se parezca mucho a lo que pasó hace 1000 símbolos, ya que el canal varía y debemos adaptarnos a dichos cambios. Y en el otro extremo, está que si se maneja una ventana pequeña no seremos capaces de extraer suficiente información de dicho conjunto para recuperar claramente la señal.

Si ahora centramos la atención sobre porqué el valor ideal de η es 1, no tenemos más que fijarnos en los experimentos 2 y 3, que indican más acertado tomar η = 0,5. Pero esta opción se ha descartado porque emplea más de cuatro horas para finalizar la prueba, respecto a usar η = 1 que necesita menos de dos horas, para en definitiva reducir el error menos del 0,2%. Así que, se escoge adoptar η = 1 como parámetro óptimo (previamente probamos sobre menores conjuntos de muestras valores de η = 2 y η = 4, y alcanzaban mayores tasas de error respecto los valores 1 y 0,5).

En cuanto al valor de β, en los experimentos 5, 6 y 7 sólo se ha variado el valor de dicha constante y se encuentra que el valor de menor error ocurre con β = 0,1. Para verificar este resultado, usamos los valores que nos proporcionan menor error con el resto de parámetros y variamos sólo β en los experimentos 20 y 21, donde vemos que para valores de β menores de 0,1 el error aumenta. Como se comprobará en el resto de canales y situaciones estudiadas, el valor de β óptimo es 0,1 puesto que al ser un valor constante y dada la similitud de los valores de los símbolos empleados en los tres canales, hace que β se comporte de forma similar en los tres ámbitos. Este hecho se demostrará, de todas formas, modificando su valor en otras pruebas que hagamos.

Conocidos ya los parámetros de mínimo error, es hora de aplicarlos sobre el resto valores de SNR estudiados para conocer las tasas de error en cada caso:

Tamaño Caché	η (Kernel)	β	Nº muestras empleadas en predicción	Tamaño ventana T	SNR	Tasa Error (%)
250	1	0,1	5	300	50	**2,5896**
250	1	0,1	5	300	30	**2,6096**
250	1	0,1	5	300	20	**2,8961**
250	1	0,1	5	300	10	**5,0716**
250	1	0,1	5	300	0	**19,638**

Tabla 2: Mejores resultados Canal 1 Perceptrón con ventana

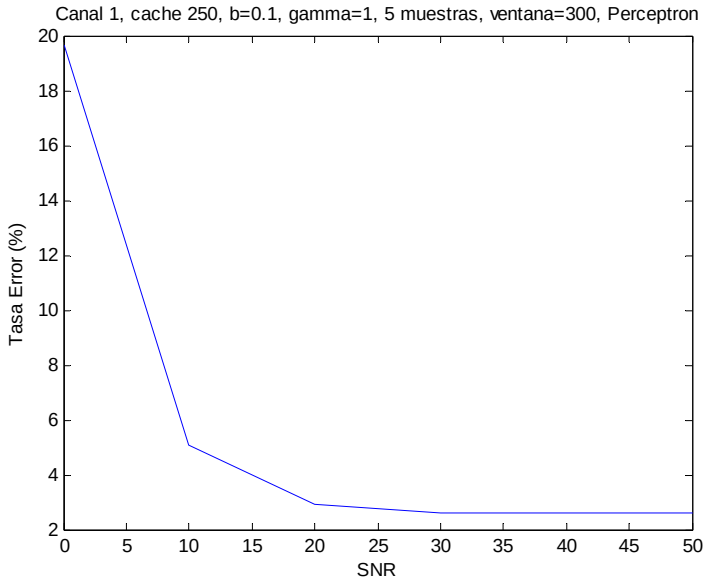

Figura 4.2.1.1 Gráfica resumen canal 1 Perceptrón con ventana

Observando la gráfica, es comprensible que a medida que señal y ruido se asemejan, la tasa de error se incremente, ya que indica que existe una peor calidad de las comunicaciones y el nivel de ruido corrompe en extremo nuestra señal obligando al igualador a trabajar lo mejor posible, pero con tasas de error cercanas al 20% para el caso SNR = 0 dB. Aun con todo, no nos parece del todo *malo* este dato, dadas las condiciones de trabajo.

Además habría que considerar la aplicación sobre la que se esté trabajando, puesto que no es lo mismo un 20% de error en una conversación telefónica que en una conexión de datos: en el primero la llamada se convertiría en algo insoportable, y en el segundo implicaría más tiempo de conexión para que los paquetes que se pierdan puedan retransmitirse, con la consiguiente implicación de estar conectado más tiempo a la red.

Sin embargo, la alta tasa de error no es del todo atribuible al igualador, ya que ante una mala situación de las comunicaciones por motivos ajenos (nivel de ruido elevado o interferencias de otros sistemas, por ejemplo) y considerando la situación de desplazamiento a altas velocidades, el error es alto pero en una situación muy específica. Vemos como con un nivel de SNR = 10 dB el error es apenas del 5%, nada que el sistema no pueda soportar en unas condiciones de trabajo no del todo deseables. Y a partir de los 20 dB, el sistema trabaja prácticamente igual que con valores de SNR mayores, demostrando que tampoco se necesitan situaciones extremadamente positivas para trabajar con resultados muy aceptables.

Si vemos otros resultados que obtuvimos para valores de SNR de 10 dB y 0 dB con distintos parámetros, las disminuciones entre el peor y mejor caso de cada uno alcanzan el 41% y 15% respectivamente:

Nº exp	Tamaño Caché	η (Kernel)	β	Nº muestras empleadas en predicción	Tamaño ventana T	SNR	Tasa Error (%)
27	75	1	0,1	4	500	10	7,268
28	**75**	**1**	**0,1**	**3**	**500**	**10**	**8,6943**
29	150	1	0,1	4	500	10	6,8105
30	150	1	0,1	6	500	10	6,3087
31	150	1	0,1	6	300	10	6,3339
32	150	1	0,1	5	300	10	5,3642
33	200	1	0,1	5	300	10	5,1871
34	**250**	**1**	**0,1**	**5**	**300**	**10**	**5,0716**

Tabla 3: Resultados Canal 1 Perceptrón con ventana 10 dB

Nº exp	Tamaño Caché	η (Kernel)	β	Nº muestras empleadas en predicción	Tamaño ventana T	SNR	Tasa Error (%)
35	**75**	**1**	**0,1**	**4**	**500**	**0**	**22,9369**
36	150	1	0,1	4	500	0	21,8092
37	**250**	**1**	**0,1**	**5**	**300**	**0**	**19,638**

Tabla 4: Resultados Canal 1 Perceptrón con ventana 0 dB

Con la gráfica siguiente apreciamos claramente, y con la misma tendencia que la anterior, la disminución del error conseguida en función de las muestras empleadas y el tamaño de la caché principal:

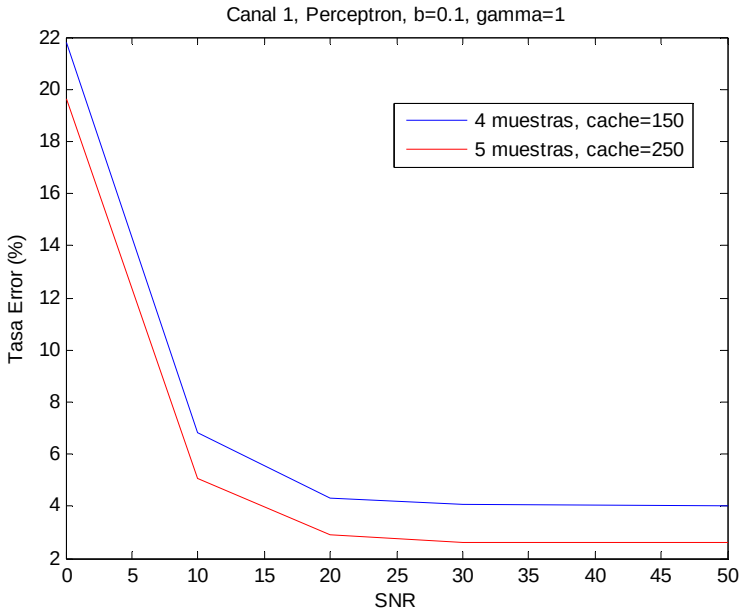

Figura4.2.1.2 Gráfica comparativa canal 1 Perceptrón con ventana

4.2.2. Canal 1 SVM sin sesgo con ventana

Los resultados obtenidos utilizando la técnica de SVM sin sesgo empleando una ventana para el caso SNR = 50 dB son:

Nº exp	Tamaño Caché	η (Kernel)	β	Nº muestras empleadas en predicción	Tam. Ventana T	Tasa Error (%)
38	75	1	0,1	4	500	4,4483
39	150	1	1	4	500	4,0724
40	150	1	0,01	4	500	4,2321
41	150	1	0,1	6	500	3,3527
42	150	1	0,1	6	250	3,1843
43	150	1	0,1	6	300	3,1332
44	**150**	**1**	**0,1**	**5**	**300**	**2,5662**

Tabla 5: Resultados Canal 1 SVM sin sesgo con ventana 50 dB

Lo único destacable de los resultados en este caso, es que obtiene unos resultados prácticamente similares a los de Perceptrón: éste tiene una tasa de error del 2,5896% y el SVM sin sesgo del 2,5662%. Una mayor complejidad matemática no demuestra una mejor respuesta frente al Perceptrón, que resulta más simple.

Con los mejores valores de los parámetros estudiados, los resultados obtenidos para SVM sin sesgo con distintos valores de SNR son:

Nº exp	Tamaño Caché	η (Kernel)	β	Nº muestras empleadas en predicción	Tamaño ventana T	SNR	Tasa Error (%)
44	250	1	0,1	5	300	50	**2,5662**

45	250	1	0,1	5	300	30	**2,6313**
46	250	1	0,1	5	300	20	**2,8996**
47	250	1	0,1	5	300	10	**5,056**
48	250	1	0,1	5	300	0	**19,664**

Tabla 6: Mejores resultados Canal 1 SVM sin sesgo con ventana

Gráficamente, vemos que se asemeja bastante al Perceptrón, tanto en los resultados finales como en la forma de la distribución del error en función de SNR.

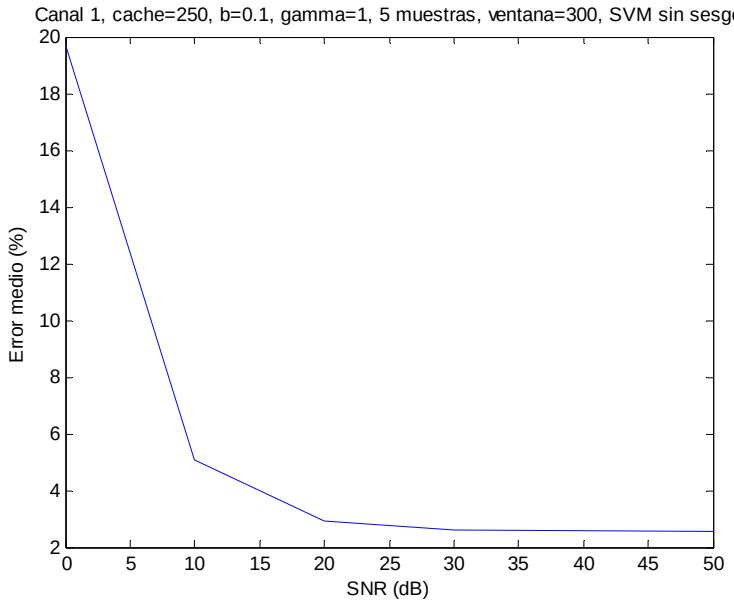

Figura 4.2.2.1 Gráfica resumen canal 1 SVM sin sesgo con ventana

Otros resultados empleando SVM sin sesgo obtenidos variando algunos parámetros para valores de SNR = 30 dB y SNR = 10 dB demuestran la disminución del error lograda entre el caso mejor y los demás:

Nº exp	Tamaño Caché	η (Kernel)	β	Nº muestras empleadas en predicción	Tamaño ventana T	SNR	Tasa Error (%)
49	75	1	0,1	4	500	30	4,4865
50	150	1	0,1	4	500	30	4,0307
51	150	1	0,1	6	500	30	3,1235
45	250	1	0,1	5	300	30	**2,6313**

Tabla 7: Resultados Canal 1 SVM sin sesgo con ventana 30 dB

Nº exp	Tamaño Caché	η (Kernel)	β	Nº muestras empleadas en predicción	Tamaño ventana T	SNR	Tasa Error (%)
52	75	1	0,1	4	500	10	7,288
53	150	1	0,1	6	500	10	6,3894
47	250	1	0,1	5	300	10	**5,036**

Tabla 8: Resultados Canal 1 SVM sin sesgo con ventana 10 dB

Las reducciones del error son considerables: con SNR = 30 dB ajustando los datos disminuimos el error respecto las consideraciones iniciales un 41%, y con 10 dB casi un 31%.

Comparando dos formas de resolución distintas gráficamente se aprecia mejor:

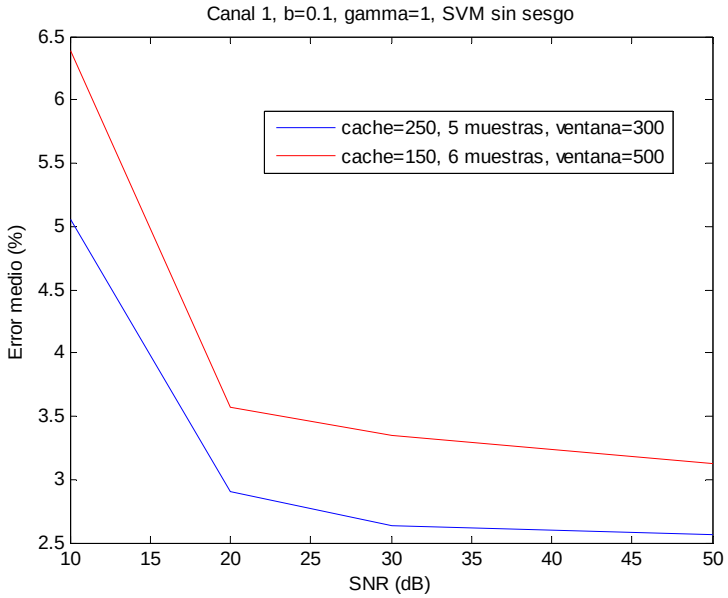

Figura 4.2.2.2 Gráfica comparativa canal 1 SVM sin sesgo con ventana

4.2.3. Canal 2 Perceptrón con ventana

Seguimos ahora con el canal 2 operando de la misma forma que antes, mostrando las tasas de error para la situación de SNR = 50 dB utilizando Perceptrón y ventana de tamaño variable.

Nº exp	Tamaño Caché	η (Kernel)	β	Nº muestras empleadas en predicción	Tam. Ventana T	Tasa Error (%)
54	75	1	0,1	4	500	1,8716
55	**150**	**1**	**0,01**	**4**	**500**	**2,053**
56	**150**	**1**	**0,001**	**4**	**500**	**2,0964**
57	150	1	0,1	6	300	1,6901
58	**150**	**1**	**0,1**	**4**	**300**	**1,4992**
59	150	1	0,1	4	200	1,4332

60	150	1	0,01	4	100	1,3603
61	150	1	0,1	4	750	1,665
62	**250**	**1**	**0,1**	**5**	**300**	**1,1797**
63	**250**	**1**	**0,1**	**4**	**300**	**1,4818**
64	**250**	**1**	**0,1**	**6**	**300**	**1,283**
65	**250**	**1**	**0,1**	**2**	**300**	**6,541**
66	250	1	0,1	5	500	1,1632
67	250	1	0,1	5	600	1,1901
68	250	1	0,1	4	500	1,612
69	**200**	**1**	**0,1**	**5**	**200**	**1,1624**
70	100	1	0,1	5	100	1,54
71	150	1	0,1	5	200	1,2153
72	250	1	0,1	5	300	1,1797
73	**200**	**1**	**1**	**5**	**200**	**0,8568**
74	200	1	2	5	200	1,2127

Tabla 9: Resultados Canal 2 Perceptrón con ventana 50 dB

Se repiten algunos patrones respecto el canal 1: a mayor caché disminuimos el error (compárense los experimentos 58 y 63) y los valores de β y η óptimos son los mismos que en el caso anterior (a menor valor de β peores resultados, tal y como se ve en las pruebas 55 y 56).

Al igual que con el canal 1 se necesitan los mismos instantes para obtener mínimo error, 5, para recuperar la señal lo más fielmente posible y reducir los efectos negativos del multitrayecto. Con ello, conocemos la cantidad de símbolos necesarios que influyen sobre los actuales reduciendo todo lo posible el error.

También se verifica que empleando menos símbolos (experimentos 63 y 65) o de más (número 64) el error sube, puesto que o no extraemos suficiente información para igualar el canal o nos pasamos, con idénticas consecuencias, aumentar el error. El experimento 62 nos sirvió de guía para saber que con 5 símbolos encontraríamos los

datos de mínimo error, y a partir de ahí sólo debíamos encontrar el tamaño de ventana adecuado.

Volviendo al hecho de usar 5 ó 6 símbolos, es cierto que entre las pruebas 64 y 69, la diferencia es de apenas el 0,12%, pero más allá de este ligero incremento, nos da una idea de la tendencia que seguirá la tasa de error si se maneja un número incorrecto de símbolos. Y puesto que tratamos de averiguar hasta dónde conseguimos disminuirlo, nos parece correcto adoptar los valores empleados en los parámetros del experimento 69 como solución final.

También vuelve a verificarse, que si forzamos la solución con determinados valores de β (como en el 73), disminuimos el error final y demostramos la robustez del modelo para resolver el problema aunque no lo escojamos como solución ideal.

En cuanto al tamaño de ventana óptimo, como no hay forma a priori de conocer el ideal, debemos ir probando con distintos tamaños en igualdad de condiciones hasta dar con la solución final. Se aprecia que a mayor tamaño a considerar el error aumenta, lo que indica que cuantas más muestras pasadas consideremos para decidir cuál eliminar de la caché principal, peores resultados obtendremos. Es decir, que las condiciones o características del canal han variado lo suficiente como para que podamos olvidarnos de ellas y nos centremos en las más próximas para igualar el canal.

De nuevo, al igual que sucedía en el canal 1, optamos por unos parámetros que no son los óptimos (si nos fijamos exclusivamente en la tasa de error), puesto que el experimento 73 con valor $\beta = 1$ logra el mejor resultado posible. Pero la razón de su descarte es la misma que el canal 1, el tiempo de realización de la prueba que existe entre ambas pruebas (cuatro horas para el número 73 frente a una del 69), obligándonos a optar por los parámetros que aportan un equilibrio entre la tasa de error obtenida y el tiempo necesario para completar el experimento.

Conocidos los mejores valores que resuelven el problema, calculamos las tasas de error para el resto de valores de SNR:

Tamaño Caché	η (Kernel)	β	Nº muestras empleadas en predicción	Tamaño ventana T	SNR	Tasa Error (%)
200	1	0,1	5	200	50	**1,1624**
200	1	0,1	5	200	30	**1,2223**

200	1	0,1	5	200	20	**1,4636**
200	1	0,1	5	200	10	**3,9558**
200	1	0,1	5	200	0	**22,294**

Tabla 10: Mejores resultados Canal 2 Perceptrón con ventana

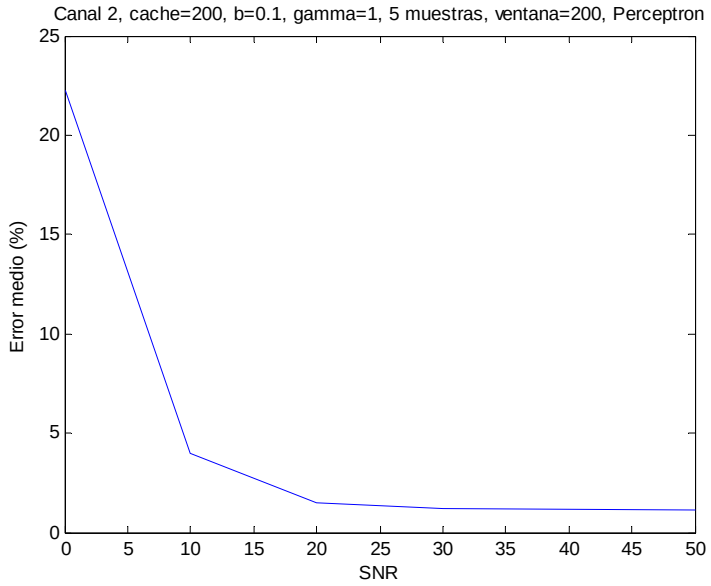

Figura 4.2.3.1 Gráfica resumen canal 2 Perceptrón con ventana

Se observan semejanzas en la distribución del error entre los canales 1 y 2 a medida que la SNR es mayor, y como incluso con un valor de 10 dB la tasa de error es inferior al 4%, demostrando otra vez su capacidad de adaptación y fortaleza en circunstancias de comunicación no muy favorables.

Otros resultados hallados para valores de SNR (30, 20, 10 y 0 dB) con otros parámetros, fueron:

Nº exp	Tamaño Caché	η (Kernel)	β	Nº muestras empleadas en predicción	Tamaño ventana T	SNR	Tasa Error (%)
75	**75**	**1**	**0,1**	**4**	**500**	**30**	**1,9263**

76	**75**	**1**	**0,1**	**3**	**500**	**30**	**3,2145**
77	**150**	**1**	**0,1**	**4**	**500**	**30**	**1,6059**
78	150	1	0,1	4	300	30	1,5582
79	**200**	**1**	**0,1**	**5**	**200**	**30**	**1,2223**
80	**200**	**1**	**1**	**5**	**200**	**30**	**0,8863**

Tabla 11: Resultados Canal 2 Perceptrón con ventana 30 dB

El experimento 80 emplea los mismos valores que el 73, que lograba el menor error con 50 dB, y aunque la disminución es significativa respecto al 79 en este caso, volvemos a encontrarnos el problema del tiempo requerido por el 80 (dos horas frente a más de siete). No merece la pena en términos absolutos optar por la solución de menor error si el tiempo requerido se dispara de esta forma.

En la siguiente tabla con SNR = 20 dB, sucede lo mismo que en casos anteriores en cuanto al tiempo necesario para acabar entre la solución adoptada (experimento 83) y la mejor (experimento 84).

Nº exp	Tamaño Caché	η (Kernel)	β	Nº muestras empleadas en predicción	Tamaño ventana T	SNR	Tasa Error (%)
81	150	1	0,1	4	500	20	2,0244
82	150	1	0,1	4	300	20	1,8664
83	**200**	**1**	**0,1**	**5**	**200**	**20**	**1,4636**
84	**200**	**1**	**1**	**5**	**200**	**20**	**1,099**

Tabla 12: Resultados Canal 2 Perceptrón con ventana 20 dB

Nº exp	Tamaño Caché	η (Kernel)	β	Nº muestras empleadas en predicción	Tamaño ventana T	SNR	Tasa Error (%)
85	**75**	**1**	**0,1**	**4**	**500**	**10**	**5,0895**

86	**150**	**1**	**0,1**	**4**	**500**	**10**	**4,5921**
87	**200**	**1**	**0,1**	**5**	**200**	**10**	**3,9558**

Tabla 13: Resultados Canal 2 Perceptrón con ventana 10 dB

Nº exp	Tamaño Caché	η (Kernel)	β	Nº muestras empleadas en predicción	Tamaño ventana T	SNR	Tasa Error (%)
88	75	1	0,1	4	500	0	24,2133
89	**200**	**1**	**0,1**	**5**	**200**	**0**	**22,294**

Tabla 14: Resultados Canal 2 Perceptrón con ventana 0 dB

Si mostramos con una gráfica la forma del error al resolver el problema de dos formas distintas, apreciamos la ligera mejoría obtenida con la solución adoptada (línea roja):

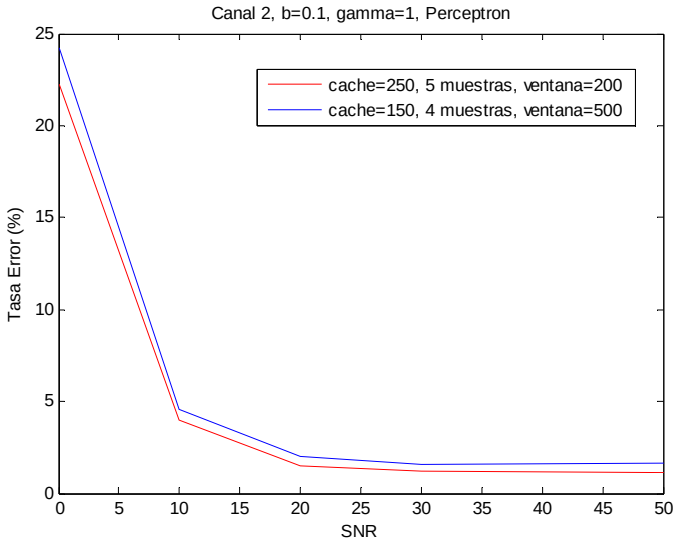

Figura 4.2.3.2 Gráfica comparativa canal 2 Perceptrón con ventana

De las tablas anteriores, podemos comparar entre ellos los experimentos 75, 76 y 77, y el 85 con el 86 para verificar que se siguen cumpliendo las conclusiones previas, respecto a los valores de tamaño de la caché, número de muestras empleadas o tamaño de la ventana.

4.2.4. Canal 2 SVM sin sesgo con ventana

Mostramos a continuación los resultados obtenidos utilizando SVM sin sesgo, comenzando por el caso de SNR = 50 dB, para averiguar cuáles son los parámetros que nos llevan al menor error.

Nº exp	Tamaño Caché	η (Kernel)	β	Nº muestras empleadas en predicción	Tam. Ventana T	Tasa Error (%)
90	75	1	1	4	500	2,2692
91	**150**	**1**	**0,1**	**4**	**500**	**1,579**
92	**150**	**1**	**0,1**	**3**	**500**	**2,902**
93	150	1	0,1	4	300	1,5131
94	150	1	0,1	4	200	1,4315
95	150	1	0,1	4	100	1,3516
96	150	1	0,1	4	50	1,382
97	**200**	**1**	**0,1**	**5**	**200**	**1,1545**
98	200	1	0,1	5	300	1,1797
99	250	1	0,1	5	300	1,1806
100	**200**	**1**	**1**	**5**	**200**	**1,4644**
101	150	1	0,1	5	200	1,2205

Tabla 15: Resultados Canal 2 SVM sin sesgo con ventana 50 dB

Los experimentos 91 y 92 comparten los mismos valores salvo el número de muestras empleadas en la predicción, donde apreciamos claramente que se reduce la tasa de error un 45% si se utilizan 4 muestras en lugar de 3. Lo que lleva a pensar en la necesidad de encontrar los símbolos necesarios para no perder capacidad de igualar el problema.

Empleando 5 símbolos, aumentando el tamaño de la caché de vectores soporte a 200 y manteniendo el tamaño de la ventana en 200, logramos el porcentaje de error más bajo (experimento 97). Necesitamos información contenida en símbolos pasados para disminuir lo posible los efectos negativos del canal e igualar el canal lo mejor posible.

Como en casos anteriores, probamos elevando el valor de la constante β hasta el valor unidad para comprobar qué resultados producía. En esta ocasión sin embargo se obtuvieron peores resultados como indica la prueba 100, descartándolo como solución.

Comparando con el Perceptrón los resultados son muy similares tanto en tasa de error como en tiempo empleado (una hora), de modo que en igualdad de condiciones parece más ventajoso elegir el Perceptrón por su mayor sencillez computacional que en casos más complejos requerirá menos tiempo. A menos que, la aplicación sea extremadamente sensible a los errores, y la pequeña diferencia que existe entre ambos métodos sea suficiente para que el sistema funcione o deje de funcionar, situación en la que SVM sin sesgo saldría ganadora.

A modo de resumen, ofrecemos los resultados alcanzados con los parámetros estudiados en la prueba 97 para otros valores de SNR y la gráfica correspondiente que refleja la evolución del error:

Tamaño Caché	η (Kernel)	β	Nº muestras empleadas en predicción	Tamaño ventana T	SNR	Tasa Error (%)
200	1	0,1	5	200	50	**1,1545**
200	1	0,1	5	200	30	**1,1936**
200	1	0,1	5	200	20	**1,4723**
200	1	0,1	5	200	10	**3,9333**
200	1	0,1	5	200	0	**22,2115**

Tabla 16: Mejores resultados Canal 2 SVM sin sesgo con ventana

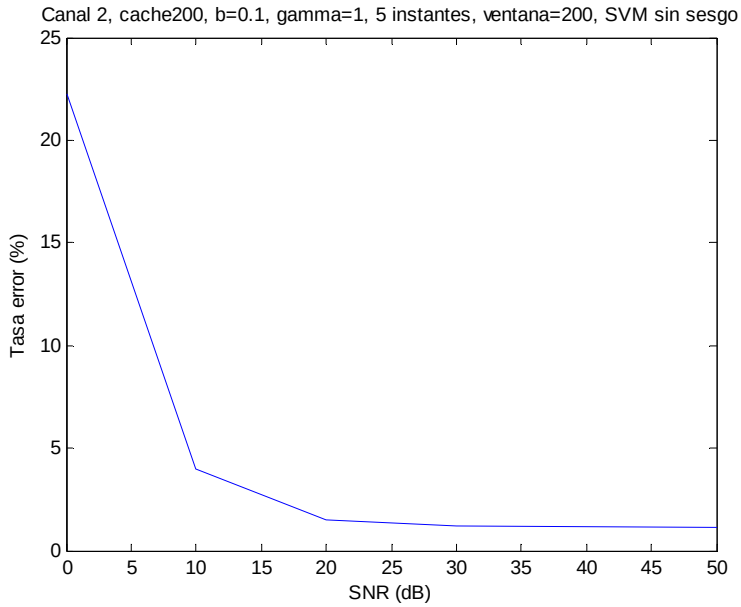

Figura 4.2.4.1 Gráfica resumen canal 2 SVM sin sesgo con ventana

Otros resultados obtenidos para distintos valores de SNR variando los valores de los parámetros arrojaron los datos siguientes:

Nº exp	Tamaño Caché	η (Kernel)	β	Nº muestras empleadas en predicción	Tamaño ventana T	SNR	Tasa Error (%)
102	75	1	0,1	4	500	30	1,915
103	150	1	0,1	4	500	30	1,632
104	**200**	**1**	**0,1**	**5**	**200**	**30**	**1,1936**

Tabla 17: Resultados Canal 2 SVM sin sesgo con ventana 30 dB

Nº exp	Tamaño Caché	η (Kernel)	β	Nº muestras empleadas en predicción	Tamaño ventana T	SNR	Tasa Error (%)
105	150	1	0,1	4	500	20	2,007
106	**200**	**1**	**0,1**	**5**	**200**	**20**	**1,4723**

Tabla 18: Resultados Canal 2 SVM sin sesgo con ventana 20 dB

Nº exp	Tamaño Caché	η (Kernel)	β	Nº muestras empleadas en predicción	Tamaño ventana T	SNR	Tasa Error (%)
107	75	1	0,1	4	500	10	5,0105
108	75	1	0,1	3	500	10	6,8379
109	150	1	0,1	4	500	10	4,685
110	**200**	**1**	**0,1**	**5**	**200**	**10**	**3,9333**

Tabla 19: Resultados Canal 2 SVM sin sesgo con ventana 10 dB

Contraponiendo los resultados de dos configuraciones distintas gráficamente, nos da una idea de la disminución del error para valores de SNR de 10, 20 y 30 dB:

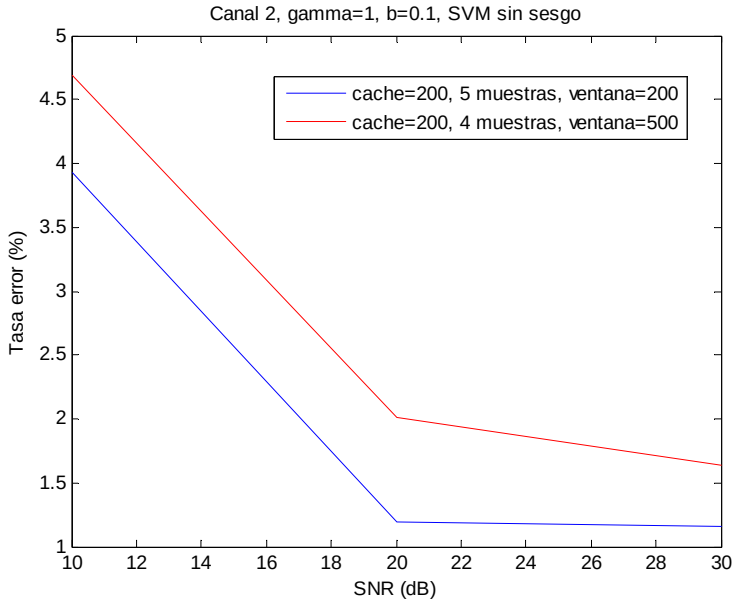

Figura 4.2.4.2 Gráfica comparativa canal 2 SVM sin sesgo con ventana

4.2.5. Canal 3 Perceptrón con ventana

Finalmente, nos falta conocer los resultados para el tercer canal que recibe 2 rayos HSDPA desplazándose a una velocidad de 50km/h. Resolviendo el problema como hasta ahora, comenzando por el caso de SNR = 50 dB, hallamos:

Nº exp	Tamaño Caché	η (Kernel)	β	Nº muestras empleadas en predicción	Tam. Ventana T	Tasa Error (%)
111	**75**	**1**	**0,1**	**4**	**500**	**9,5134**
112	75	1	0,1	3	500	8,4317
113	150	1	0,01	4	500	8,7269
114	150	1	0,1	3	500	8,2468
115	**150**	**1**	**0,1**	**6**	**500**	**8,2095**
116	150	1	0,1	3	100	8,4907
117	150	1	0,1	3	300	8,9135
118	150	1	0,1	3	700	9,8345
119	150	1	0,1	2	200	9,3267
120	150	**1**	0,1	**2**	**300**	**6,8457**
121	150	1	0,1	2	400	8,8571
122	150	1	1	2	300	7,7876
123	150	1	0,01	2	300	7,9881
124	**150**	**1**	**0,1**	**2**	**300**	**9,3339**
125	150	1	1	2	500	9,1896
126	150	1	0,1	2	500	8,5611
127	**250**	**1**	**0,1**	**2**	**300**	**8,1348**
128	150	1	0,5	2	300	8,7191
129	150	1	2	2	300	7,2945
130	**150**	**1**	**0,2**	**2**	**300**	**9,4856**

Tabla 20: Resultados Canal 3 Perceptrón con ventana 50 dB

El canal 3 es el que ofrece, sin duda, los resultados más controvertidos de todos y del que se pueden extraer las conclusiones más relevantes, pues para el mejor caso posible la tasa de error es casi del 7%, muy superior a las de los canales 1 y 2. Y se puede apreciar cómo se han usado multitud de valores para intentar encontrar las menores tasas de error, sin lograr descender del alcanzado en la prueba 108.

De las cuestiones menos relevantes están que las constantes η y β mantienen la tendencia de casos anteriores en cuanto a valores óptimos, y que elevar el valor de β por encima de 0,1 provoca que empeoremos el error y aumentemos el tiempo requerido para completar el experimento. La mejor prueba de ello, se tiene comparando los números 120 y 130, en los que la única diferencia es el valor de β (0,1 y 0,2 respectivamente) y el error crece casi un 30%, y en términos absolutos de error, no es lo mismo 9,5% que 6,8%, aun cuando un 6,8 nos parece bastante alto para ser el caso mejor encontrado. Tampoco detallaremos el hecho de que aumentando el tamaño de la caché principal reduzcamos el error final (experimentos 124 y 127), puesto que podemos remitirnos a lo visto en casos anteriores.

La conclusión inmediata que podemos sacar es que dadas las características de este canal que trabaja únicamente con 2 rayos HSDPA a 50 km/h, es que habiendo probado distinta cantidad de muestras para igualar el canal, parece indicar que aun en el mejor caso hallado, las tasas de error son sensiblemente peores respecto los otros dos canales. La cantidad de rayos HSDPA que se reciben pueden ser una de las causas de estas tasas de error más elevadas, ya que si no recibimos el número suficiente de rayos la igualación de canal se ve muy comprometida y no somos capaces de resolverlo aun empleando mayor número de símbolos como atestiguan los experimentos 111 y 115. Esto se deberá al hecho de que los símbolos están poco o nada correlacionados entre ellos y aunque tratemos de coger muestras pasadas no sirven para predicciones actuales.

Es decir, que tal y como se ha planteado la resolución del problema, para superar el multitrayecto a altas velocidades con ciertas garantías, requerimos de un número suficiente de reflexiones (rayos) como se vio en los canales 1 y 2.

Conocidos entonces los parámetros de mínimo error, los extrapolamos para el resto de valores de SNR estudiados:

Tamaño Caché	η (Kernel)	β	Nº muestras empleadas en predicción	Tamaño ventana T	SNR	Tasa Error (%)
150	1	0,1	2	300	50	**6,8457**
150	1	0,1	2	300	30	**9,3397**
150	1	0,1	2	300	20	**11,2877**
150	1	0,1	2	300	10	**13,418**
150	1	0,1	2	300	0	**29,2007**

Tabla 21: Mejores resultados Canal 3 Perceptrón con ventana

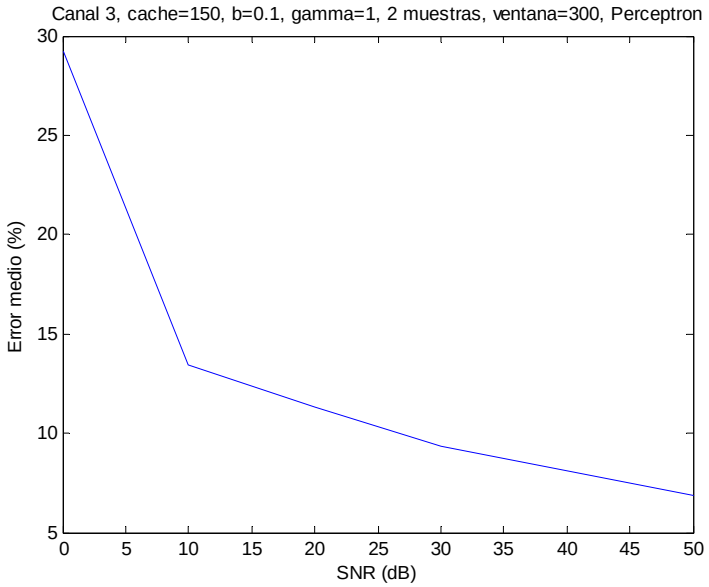

Figura 4.2.5.1 Gráfica resumen canal 3 Perceptrón con ventana

4.2.6. Canal 3 SVM sin sesgo con ventana

Usando SVM sin sesgo, para SNR = 50 dB se obtiene:

Nº exp	Tamaño Caché	η (Kernel)	β	Nº muestras empleadas en predicción	Tam. Ventana T	Tasa Error (%)
131	75	1	1	4	500	9,1071
132	75	1	0,1	4	500	9,2382
133	150	1	0,1	4	500	9,1514
134	75	1	0,1	3	500	9,3831
135	150	1	0.1	2	300	10,1696
136	**150**	**1**	**0,1**	**2**	**500**	**8,2416**
137	150	1	0,1	2	200	9,273

Tabla 22: Resultados Canal 3 SVM sin sesgo con ventana 50 dB

En esta situación, usar SVM sin sesgo no resulta tan apropiado como el Perceptrón, ya que la menor tasa alcanzada es casi un 1,5% superior. Y puesto que no sucede como en los casos anteriores en que parecía indiferente usar una u otra técnica, hemos descartado calcular la tasa de error que se obtendría para otros valores de SNR, dado que en el mejor caso posible es sensiblemente peor.

4.2.7. Canal 1 Perceptrón sin ventana

Usamos ahora el programa que elimina el vector soporte más antiguo de la caché principal y probamos de nuevo sobre los tres canales y las dos técnicas estudiadas.

Comenzamos mostrando los resultados para el canal 1 con Perceptrón y valor SNR = 50 dB con distintos parámetros, y dado que contamos con la experiencia previa de los experimentos desarrollados con la otra variante, probamos con los valores que nos otorgaban mínimo error para comparar ambas situaciones en igualdad de condiciones.

Nº exp	Tamaño Caché	η (Kernel)	β	Nº muestras empleadas en predicción	Tasa Error (%)
1	**150**	**1**	**0,1**	**6**	**3,3423**
2	**250**	**1**	**0,1**	**5**	**2,6461**
3	**250**	**1**	**1**	**5**	**1,9715**
4	**200**	**1**	**0,1**	**5**	**2,6886**
5	300	1	0,1	5	2,699

Tabla 23: Resultados Canal 1 Perceptrón sin ventana 50 dB

De nuevo nos encontramos con el hecho de que la constante β condiciona el resultado en gran medida, puesto que el experimento 3 logra el mejor resultado alcanzado, pero a costa de necesitar cerca de siete horas respecto el experimento 2 que requiere dos horas. Los valores de este último serán los que usemos para calcular el error sobre el resto de valores de SNR.

De la constante η y del número de muestras empleadas en la predicción, no añadimos nada nuevo de lo visto hasta ahora, ya que sólo se varía la forma de aligerar la caché de vectores soporte, mientras que el resto de cálculos necesarios para igualar el canal usando *kernel* son iguales tanto si se usa o no la ventana. Y ya vimos que con 5 símbolos se obtenían los mejores resultados para el canal 1.

El cuadro resumen y su gráfica para el canal 1 son:

Nº exp	Tamaño Caché	η (Kernel)	β	Nº muestras empleadas en predicción	SNR (dB)	Tasa Error (%)
2	250	1	0,1	5	50	**2,6461**
3	250	1	0,1	5	30	**2,6513**
4	250	1	0,1	5	20	**2,9404**
5	250	1	0,1	5	10	**5,1741**
6	250	1	0,1	5	0	**19,8455**

Tabla 24: Mejores resultados Canal 1 Perceptrón sin ventana

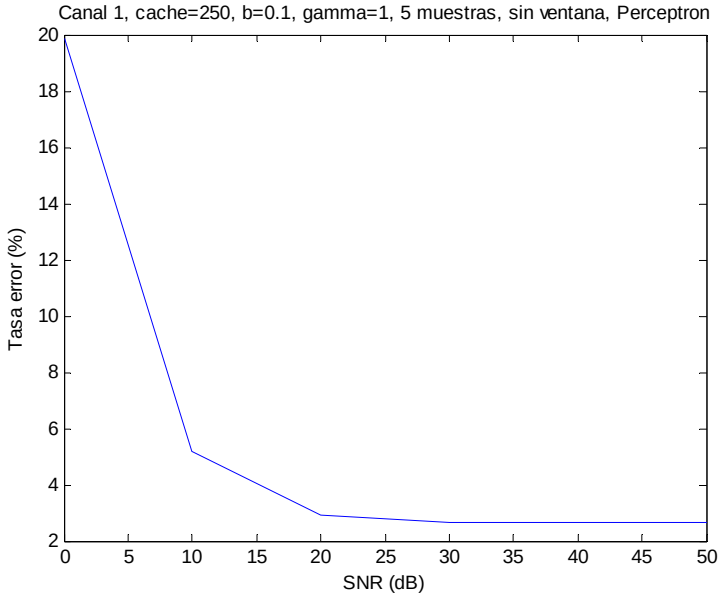

Figura 4.2.7.1 Gráfica resumen canal 1 Perceptrón sin ventana

Con los siguientes datos comparamos entre distintos tamaños de caché y número de símbolos empleados para atestiguar la constatación de la disminución de error lograda:

Nº exp	Tamaño Caché	η (Kernel)	β	Nº muestras empleadas en predicción	SNR (dB)	Tasa Error (%)
7	150	1	0,1	6	30	3,3597
3	250	1	0,1	5	30	**2,6513**
8	150	1	0,1	6	20	3,6314
4	250	1	0,1	5	20	**2,9404**
9	150	1	0,1	6	10	6,7315
5	250	1	0,1	5	10	**5,1741**

Tabla 25: Otros resultados Canal 1 Perceptrón sin ventana

Utilizando un símbolo de más en los cálculos y comparando los resultados alcanzados, se aprecia de nuevo que sólo usando el número correcto de muestras en los cálculos el error mejora notablemente: entre un 20-25% de media para diversos valores de SNR. Si lo vemos gráficamente:

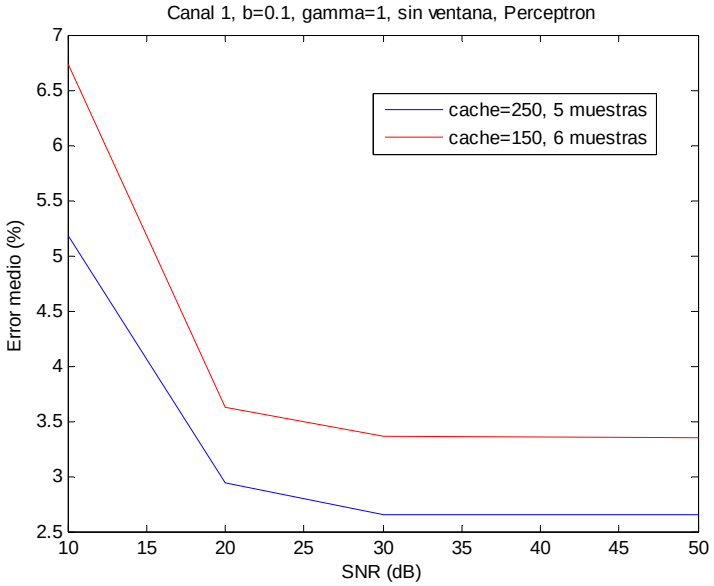

Figura 4.2.7.2 Gráfica comparativa canal 1 Perceptrón sin ventana

4.2.8. Canal 1 SVM sin sesgo sin ventana

Empleando SVM sin sesgo, los resultados obtenidos y los comentarios que podemos extraer son prácticamente similares respecto al caso anterior tal y como se puede ver:

Nº exp	Tamaño Caché	η (Kernel)	β	Nº muestras empleadas en predicción	SNR (dB)	Tasa Error (%)
10	150	1	0,1	6	50	3,3519
11	150	1	0,1	6	30	3,4022

12	150	1	0,1	6	20	3,6791
13	150	1	0,1	6	10	6,7853
14	**250**	**1**	**0,1**	**5**	**50**	**2,6018**
15	**250**	**1**	**0,1**	**5**	**30**	**2,706**
16	**250**	**1**	**0,1**	**5**	**20**	**2,9421**
17	**250**	**1**	**0,1**	**5**	**10**	**5,2331**
18	**250**	**1**	**0,1**	**5**	**0**	**19,842**

Tabla 26: Resultados Canal 1 SVM sin sesgo sin ventana

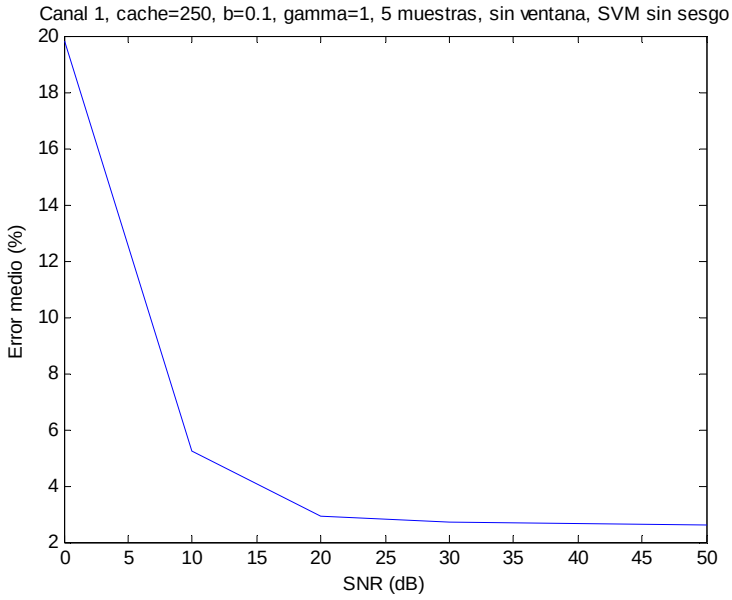

Figura 4.2.8.1 Gráfica resumen canal 1 SVM sin sesgo sin ventana

Se cumple al igual que en casos anteriores, que tanto Perceptrón como SVM sin sesgo alcanzan tasas de error comparables.

4.2.9. Canal 2 Perceptrón sin ventana

Para el caso del canal 2 con Perceptrón y SNR = 50 dB realizamos las siguientes pruebas:

Nº exp	Tamaño Caché	η (Kernel)	β	Nº muestras empleadas en predicción	SNR (dB)	Tasa Error (%)
19	150	1	0,1	4	50	1,4054
20	**250**	**1**	**0,1**	**5**	**50**	**1,2466**
21	250	1	0,1	6	50	1,3186
22	350	1	0,1	5	50	1,382
23	300	1	0,1	5	50	1,29

Tabla 27: Resultados Canal 2 Perceptrón sin ventana 50 dB

Vemos de nuevo que gracias al cambio de estrategia usada en estos experimentos, podemos ampliar el tamaño de la caché principal en 100 unidades respecto al tamaño máximo que utilizamos con la ventana deslizante. Esto se debe al hecho de que al no tener que averiguar qué vector soporte provoca menos errores al retirarlo de la caché principal, sin más que eliminar el más antiguo, evitamos hacer una gran cantidad de operaciones que consumen mucha memoria.

Aun con todo, lo destacable es que aunque aumentemos el tamaño de la caché principal no nos lleva a conseguir menores tasas de error en este escenario. Es decir, que si consideramos muestras demasiado alejadas de la actual no mejoramos la igualación, a causa de las variaciones existentes en el canal que provocan diferencias notables entre muestras demasiado alejadas en el tiempo.

En cuanto a los valores del resto de parámetros no existen diferencias destacables de lo visto hasta ahora.

Sí es de apreciar la fortaleza del algoritmo para la situación de valores de SNR especialmente bajos como en los 10 dB, con menos del 4% de error.

El resumen con los resultados para los valores de SNR estudiados con los mejores parámetros y la consiguiente gráfica, son:

Nº exp	Tamaño Caché	η (Kernel)	β	Nº muestras empleadas en predicción	SNR (dB)	Tasa Error (%)
20	250	1	0,1	5	50	**1,2466**
24	250	1	0,1	5	30	**1,3143**
25	250	1	0,1	5	20	**1,533**
26	250	1	0,1	5	10	**3,9168**
27	250	1	0,1	5	0	**22,0926**

Tabla 28: Mejores resultados Canal 2 Perceptrón sin ventana

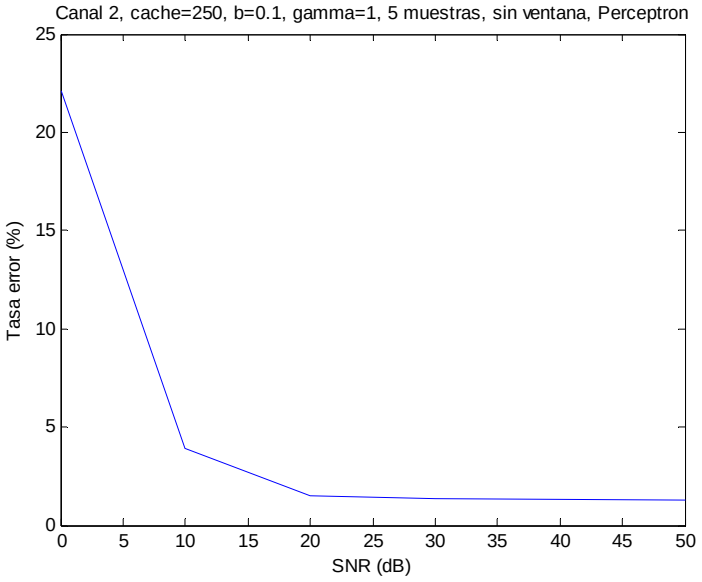

Figura 4.2.9.1 Gráfica resumen canal 2 Perceptrón sin ventana

4.2.10. Canal 2 SVM sin sesgo sin ventana

Si usamos ahora SVM sin sesgo, con los mismos parámetros que en el caso anterior (ya que los valores óptimos se repiten según el tipo de canal estudiado después de verificarlo con algunas pruebas extras), los resultados finales son:

Nº exp	Tamaño Caché	η (Kernel)	β	Nº muestras empleadas en predicción	SNR (dB)	Tasa Error (%)
28	250	1	0,1	5	50	**1,2662**
29	250	1	0,1	5	30	**1,2943**
30	250	1	0,1	5	20	**1,5495**
31	250	1	0,1	5	10	**3,8977**
32	250	1	0,1	5	0	**22,2332**

Tabla 29: Mejores resultados Canal 2 SVM sin sesgo sin ventana

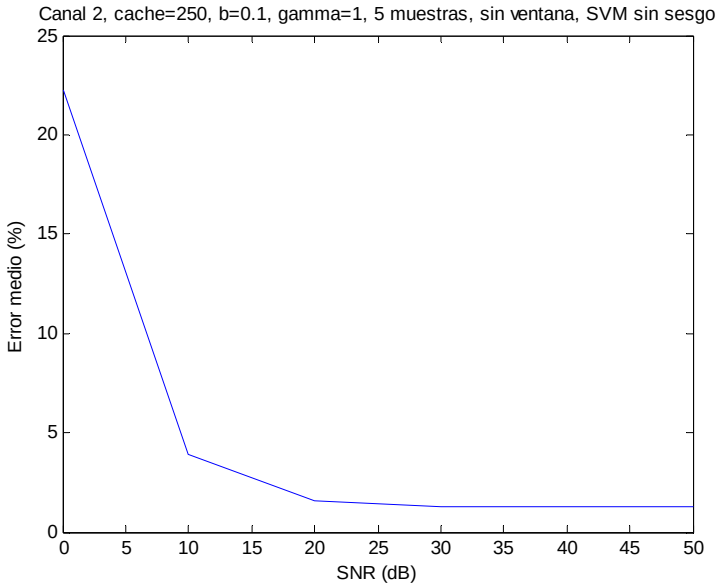

Figura 4.2.10.1 Gráfica resumen canal 2 SVM sin sesgo sin ventana

Comparando los resultados con los obtenidos anteriormente por el Perceptrón, éste consigue en términos generales una ligera disminución en los casos extremos estudiados de 50 y 0 dB.

4.2.11. Canal 3 Perceptrón sin ventana

Para valores de SNR = 50 dB con el canal 3 usando Perceptrón, obtuvimos valores menos homogéneos que en casos anteriores:

Nº exp	Tamaño Caché	η (Kernel)	β	Nº muestras empleadas en predicción	SNR (dB)	Tasa Error (%)
33	**150**	**1**	**0,1**	**3**	**50**	**8,3805**
34	**150**	**1**	**0,1**	**2**	**50**	**7,5141**
35	**250**	**1**	**0,1**	**2**	**50**	**6,4308**
36	300	1	0,1	2	50	6,3527
37	**350**	**1**	**0,1**	**2**	**50**	**5,9177**
38	300	1	0,1	3	50	7,2763

Tabla 30: Resultados Canal 3 Perceptrón sin ventana 50 dB

En el canal 3 se da el hecho de que aumentando en 100 muestras la caché principal respecto a los valores máximos que veníamos usando hasta ahora, logramos reducir el error final (comparando experimentos 34, 35 y 37).

Lo más destacable se aprecia contrastando los experimentos 33 y 34, que sólo difieren en la cantidad de símbolos que emplean en los cálculos de predicción. Se produce la misma situación que se estudió con la ventana deslizante, con dos símbolos se obtienen las tasas más bajas de error, y si se utilizan más no nos aporta nueva

información al conjunto, haciendo que se empeore la apreciación que podamos tener sobre las muestras que manejemos próximamente y se eleve el error.

Por último, destacar también que en esta situación y con la estrategia de eliminar directamente el vector soporte más antiguo, aumentando la caché principal hasta 350 logramos menor error. Al desplazarnos a menor velocidad en este canal, con un mayor conjunto de vectores soporte descendemos la tasa de error, gracias a que los vectores soporte almacenados son muy representativos del conjunto y se requiere más cantidad para que al eliminar el más antiguo mantengamos un error bajo.

Los mejores parámetros estudiados nos reportan una tasa de error final para los diferentes valores de SNR estudiados de:

Nº exp	Tamaño Caché	η (Kernel)	β	Nº muestras empleadas en predicción	SNR (dB)	Tasa Error (%)
37	350	1	0,1	2	50	**5,9177**
39	350	1	0,1	2	30	**6,5228**
40	350	1	0,1	2	20	**8,6149**
41	350	1	0,1	2	10	**14,3451**
42	350	1	0,1	2	0	**31,2632**

Tabla 31: Mejores resultados Canal 3 Perceptrón sin ventana

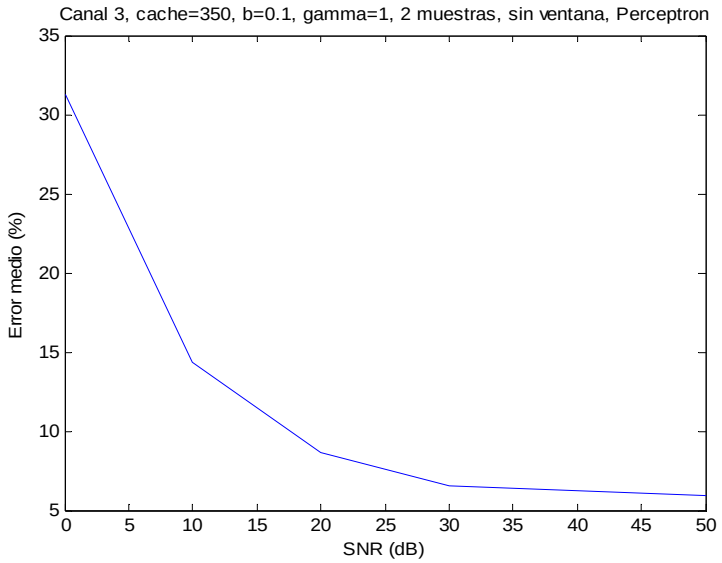

Figura 4.2.11.1 Gráfica resumen canal 3 Perceptrón sin ventana

4.2 12. Canal 3 SVM sin sesgo sin ventana

Conocidos los valores para los que obtenemos el mínimo error usando Perceptrón según los resultados anteriores, verificamos con un par de ejemplos que efectivamente tales valores constituyen los de mínimo error usando ahora SVM sin sesgo. Los resultados como podemos notar son prácticamente similares respecto el caso previo.

Nº exp	Tamaño Caché	η (Kernel)	β	Nº muestras empleadas en predicción	SNR (dB)	Tasa Error (%)
43	150	1	0,1	3	50	8,4482
44	150	1	0,1	3	30	8,5385
45	**350**	**1**	**0,1**	**2**	**50**	**6,0558**
46	250	1	0,1	2	50	6,2363
47	**350**	**1**	**0,1**	**2**	**30**	**6,4785**
48	**350**	**1**	**0,1**	**2**	**20**	**8,8111**
49	**350**	**1**	**0,1**	**2**	**10**	**14,3885**
50	**350**	**1**	**0,1**	**2**	**0**	**31,2077**

Tabla 32: Resultados Canal 3 SVM sin sesgo sin ventana

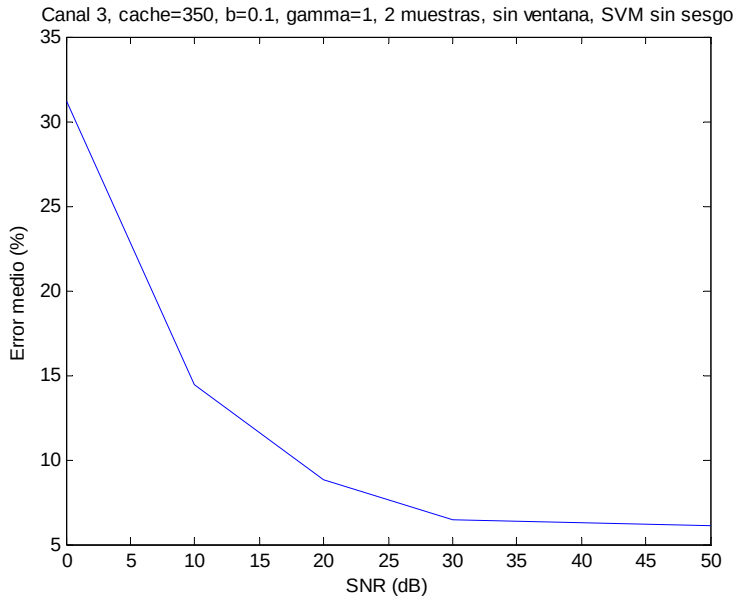

Figura 4.2.12.1 Gráfica resumen canal 3 SVM sin sesgo sin ventana

4.2.13. Resumen canales

A modo de resumen ofrecemos los parámetros que consiguen la tasa de error más baja para cada canal estudiado, y una gráfica resumen que enfrenta a los tres canales.

Canal	Tamaño Caché	η (Kernel)	β	Nº muestras empleadas en predicción	Tam. Ventana T	Técnica	Tasa Error (%)
1	150	1	0,1	5	300	SVM	2,5662
2	200	1	0,1	5	200	SVM	1,1545
3	350	1	0,1	2	No	Percep.	5,9177

Tabla 33: Resumen mejores parámetros por canales

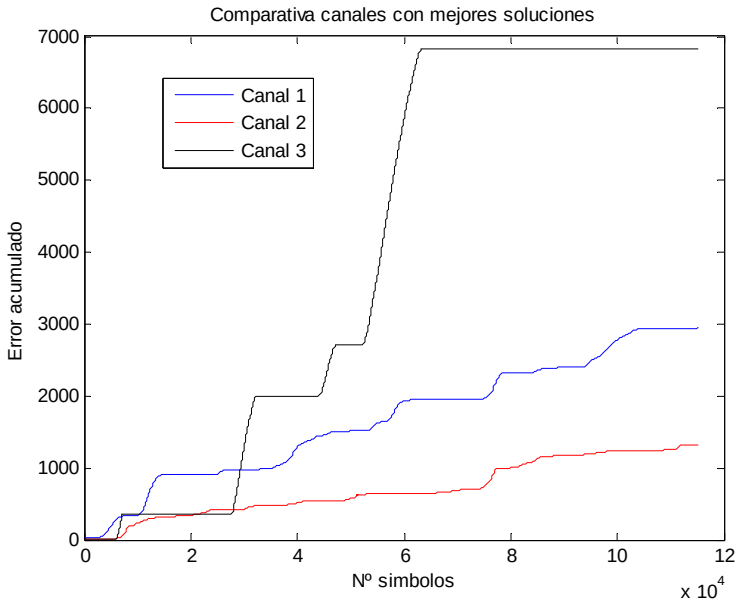

Figura 4.2.13.1 Gráfica comparativa entre los tres canales

4.3. Guiado por decisión: comparación de resultados

En lugar de repetir todos los experimentos realizados anteriormente variando el hecho de que no todos los símbolos sean pilotos, sino sólo 45 tramas de 512 símbolos equiespaciados a lo largo de la trama total de $115.2 \cdot 10^3$ símbolos (cada 2560 símbolos tenemos 512 pilotos), utilizaremos los valores de los parámetros que han ocasionado menor error en los casos antes estudiados para valores de SNR de 50 y 10 dB para poder comparar los resultados usando todos los símbolos como pilotos o sólo una pequeña parte, aplicando guiado por decisión en este caso.

Se han agrupado los resultados de dos en dos en función de la SNR, para comparar más fácilmente las tasas cuando todos los datos son pilotos y cuando no.

¿Todo pilotos?	Canal	Técnica	SNR (dB)	Error (%)
Sí	1	Percep.	50	**2,5896**
No	1	Percep.	50	**7,802**
Sí	1	Percep.	10	**5,0716**
No	1	Percep.	10	**11,8127**
Sí	2	Percep.	50	**1,1624**
No	2	Percep.	50	**5,8847**
Sí	2	Percep.	10	**3,9558**
No	2	Percep.	10	**12,509**
Sí	3	Percep.	50	**6,8457**
No	3	Percep.	50	**9,8597**
Sí	3	Percep.	10	**13,418**
No	3	Percep.	10	**15,2653**
Sí	1	SVM	50	**2,5662**
No	1	SVM	50	**8,348**
Sí	1	SVM	10	**5,056**
No	1	SVM	10	**12,0028**
Sí	2	SVM	50	**1,1545**
No	2	SVM	50	**5,1825**
Sí	2	SVM	10	**3,9333**
No	2	SVM	10	**11,9673**
Sí	3	SVM	50	**8,2416**
No	3	SVM	50	**10,6488**

Tabla 34: Comparación guiado por decisión vs todo pilotos

Con la nueva situación en la que trabajamos con y sin pilotos a lo largo de la trama de datos, la resolución y planteamiento del problema sigue siendo la misma. Sólo varía ahora el hecho de que si el símbolo no es piloto, tenemos que comparar el signo del símbolo tratado con el original, y cada vez que no coincidan, se produce un error y el sistema tiene que corregirlo y realizar los cálculos tal y como se vio al presentar los algoritmos estudiados. Mientras que los pilotos sirven para sincronizarnos con la trama de datos si en determinadas partes se ha acumulado gran cantidad de error, ya que al

trabajar con el símbolo original nos ayuda a que los siguientes cálculos que se realicen sean con los valores correctos. Luego está en manos del algoritmo ser capaz de igualar el canal lo mejor posible, tal y como hemos visto hasta ahora.

Como podemos apreciar en los resultados, una vez que trabajamos en un escenario real el error aumenta respecto el caso ideal en todos los casos. A pesar de que en porcentajes los incrementos son muy considerables, en términos absolutos, y tomando los resultados con cierta cautela puesto que nuestro planteamiento es más teórico que práctico, obtenemos unos niveles de error aceptables para los casos y situaciones estudiadas, como puede verse para los canales 1 y 2 con tasas de entre el 5-8%.

El punto favorable a considerar es que a partir del nivel de los 20 dB de SNR, las tasas de error en los casos teóricos eran prácticamente similares entre ellos, lo que nos hace suponer que a partir de tales niveles de SNR las tasas de error serán similares a las halladas con 50 dB.

Podemos ilustrar con unas cuantas gráficas la evolución del error a lo largo del problema viendo como se acumula el error cometido, comparando los peores escenarios para apreciar mejor el comportamiento en ambas situaciones estudiadas: se aprecia que la tendencia es similar en ambos problemas, sólo que ahora vamos acumulando más error a lo largo de la trama de datos.

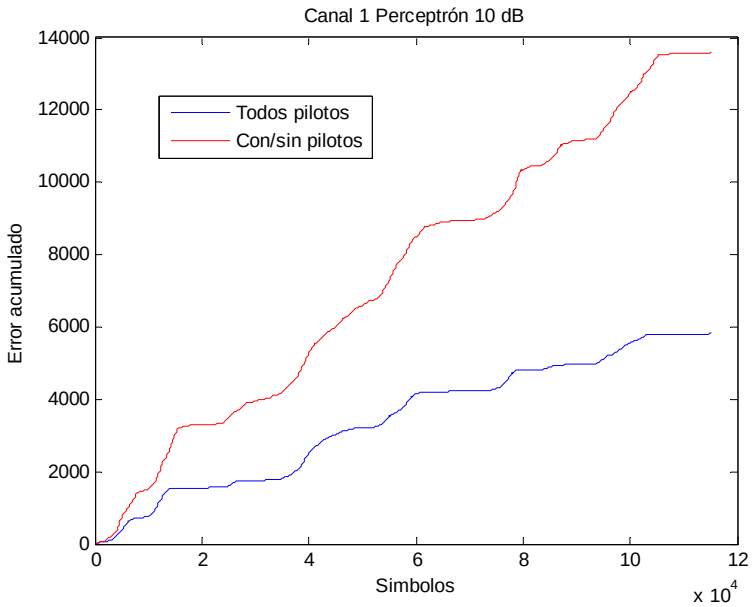

Figura 4.3.1 Comparación todo pilotos vs guiado por decisión canal 1

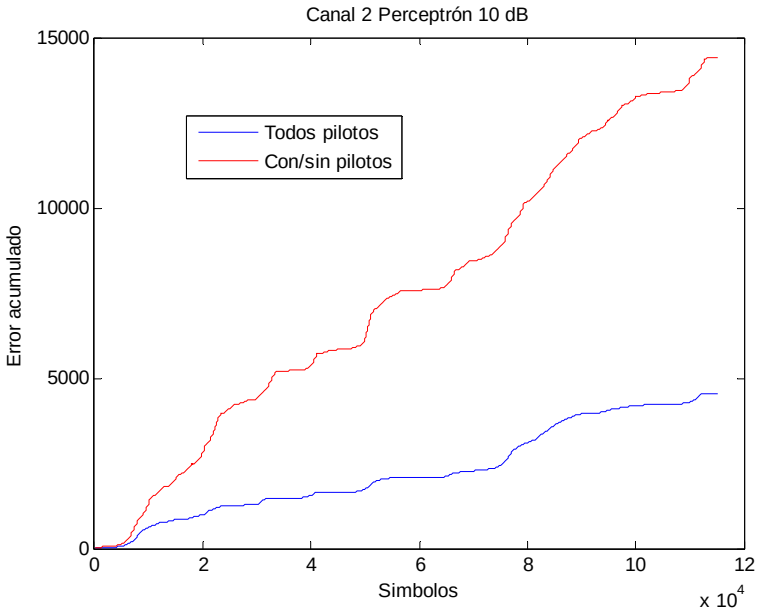

Figura 4.3.2 Comparación todo pilotos vs guiado por decisión canal 2 Perceptrón

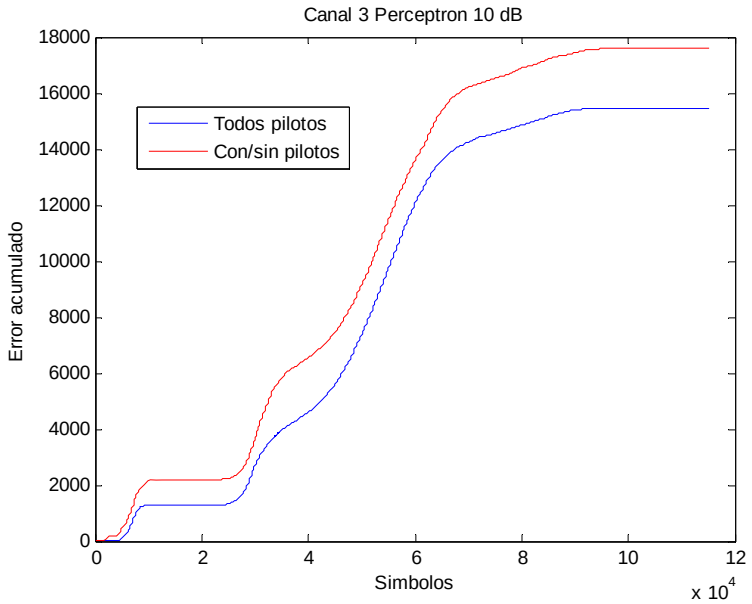

Figura 4.3.3 Comparación todo pilotos vs guiado por decisión canal 3

Capítulo 5
Conclusiones

Las conclusiones que podemos extraer de nuestro proyecto se resumen en:

- En el uso de las técnicas de Perceptrón o SVM, parece claro que debido a la igualdad de las tasas de error obtenidas el Perceptrón parece más idóneo por el hecho de resultar más sencillo matemáticamente, como ya hemos apuntado.
- Manejando un tamaño de caché de vectores soporte suficientes, el error disminuye hasta su tope máximo, pues con un conjunto representativo será más fácil averiguar qué vector soporte incomoda menos si lo quitamos como solución para poder incorporar otros a la caché cuando corresponda.
- Prácticamente lo mismo sucede con el tamaño de la ventana deslizante, tiene que tener un tamaño determinado que garantice una buena representatividad del conjunto para conseguir bajas tasas de error, pues con grupos muy grandes se dispersan los resultados debido a que muestras muy previas (ventana grande) apenas se parecen a las presentes, y con pocas (ventana pequeña) no somos capaces de extraer toda la información necesaria para hacernos una idea de las siguientes.
- En lo referente a los dos algoritmos utilizados, entre eliminar el vector soporte más antiguo o aquel que ocasione menor error entre un conjunto determinado de muestras, para los canales 1 y 2 los resultados no difieren en exceso entre uno y otro. El hecho de que sea algo menor al utilizar una ventana concreta se debe precisamente al hecho de que matemáticamente resulta más exacto saber sobre qué muestras se debe comprobar el error cometido para eliminar la más oportuna en cada momento, mientras que la otra aplica una estrategia más conservadora. Optando por una de las dos, la complejidad computacional parece decidir en favor de la técnica de eliminación, pues implica menos cálculos, por lo que sería la modalidad escogida para resolver el problema tanto en la situación de todos pilotos o no.

- Que las tasas de error sean mejores en el caso del canal 2, nos indica que es el que posee mejores características de configuración en cuanto al equilibrio entre velocidad y número de rayos HSDPA. Los dos rayos de más que recibe el canal 1 sólo provocan que el error sea más del doble para el canal 1 respecto el canal 2. Pero siendo justos, diremos que los resultados para ambos canales son del todo satisfactorios, puesto que para un valor de SNR = 10 dB el máximo error es del 5%, que nos parece muy asumible dadas las condiciones de trabajo simuladas.

- Con el canal 3 sí podíamos suponer que no habíamos averiguado los valores de los parámetros que nos llevan al menor error, pero una vez que realizamos las pruebas con y sin pilotos y vimos que las tasas de error eran comparables a las de los otros canales, comprendimos que en la situación ideal con todos pilotos el factor de trabajar sólo con dos rayos HSDPA no es suficiente para que el sistema iguale el canal de una manera del todo aceptable.

- De los parámetros que definen los canales, velocidad de desplazamiento y rayos HSDPA que recibimos, ha quedado patente que el número de rayos es más importante que el de la velocidad, ya que sin un número suficiente de ellos las técnicas aplicadas no logran buenos resultados y el problema del multitrayecto afecta notablemente provocando mayores tasas de error.

- Debemos recordar que aplicamos la solución de un clasificador lineal, del que ya anunciamos no ofrecía la mejor solución posible pero sí mas más sencilla, y los canales manejados no han sido creados por nosotros. Por ello, las tasas de error en el caso de trabajar con y sin pilotos con tales condiciones de trabajo, no nos parecen del todo excesivas.

- Sin duda el mayor causante de que el tiempo sea excesivo en algunas pruebas se debe al hecho de que la constante β posea un valor lo suficientemente grande, como para que la mayoría de los símbolos tengan un valor de predicción menor que β. Esto lo que hace es que a una gran cantidad de símbolos se le considera vector soporte, se incluya en la caché, y cuando esté completa y tengamos que eliminar según la técnica escogida, se requiere gran cantidad de memoria en operaciones complejas con la caché y el *kernel* que incrementan sustancialmente el tiempo de procesamiento.

- El hecho de descartar algunos experimentos a causa del tiempo empleado se debe al hecho de que las condiciones en que se realizaron las pruebas, el entorno de desarrollo y los algoritmos eran iguales para todos.

- Queda patente entonces, que nuestro planteamiento inicial de resolver el problema de igualación de canal en entornos estacionarios como otro problema alternativo para verificar si seguimos los cambios que se producen en el canal con algoritmos de aprendizaje máquina es factible, puede resolverse y consigue tasas de error asequibles (desde el punto de vista de aprendizaje máquina) en determinadas situaciones considerando el conjunto de muestras y condiciones que hemos manejado y supuesto en el problema. El hecho de empeorar (pero manteniendo tasas de error aceptables para el aprendizaje máquina, y recordando que no pretendemos competir con los algoritmos utilizados en igualación de canal) en la situación de trabajar con y sin pilotos, abre la posibilidad de continuar esta línea de investigación que aquí hemos iniciado mediante mejoras en la forma de utilizar el *kernel*, ampliar el rango de valores de los parámetros empleados, otro lenguaje de programación, otras técnicas distintas de Perceptrón o SVM, o utilizar algoritmos más actuales.

Capítulo 6

Presupuesto

Equipo necesario para realizar el proyecto

- Material de oficina, tinta impresora…….. ……25€
- Hardware: ordenador con 1 Gb memoria RAM, 60 Gb HD, teclado, ratón, pantalla TFT 19''…… ……..350€
- Software: S.O. XP Professional, Matlab v7, Open Office……6000€
- Gasto energético 1200 horas x 0,055075 kW/h…….66€
- Conexión a internet 20€ x 4 meses……80€
- Mano de obra 833 horas x 22€/hora……18326€

TOTAL……**24847€**

Capítulo 7

Referencias y Bibliografía

Referencias

[1] J. Weston, A. Bordes, L. Bottou, *Online (and Offline) on an Even Tighter Budget*. Proceedings of the Tenth International Workshop on Artificial Intelligence and Statistics, 2005.

[2] http://hspa.gsmworld.com/

[3] Antonio Artés Rodríguez, Fernando Pérez González, *Comunicaciones Digitales*. Editorial Pearson Educación, 2007

[4] Asignatura *Fundamentos de comunicaciones móviles*, Universidad Carlos III de Madrid.

[5] Marcelino Lázaro, *Comunicaciones Digitales Avanzadas: Diseño de Receptores de Comunicaciones*.

[6] Asignatura *Tratamiento Digital de la Información*, Universidad Carlos III de Madrid.

[7] Nello Cristianini, John Shawe-Taylor, *An introduction to Support Vector Machines.* Editorial Cambridge University Press, 2000

[8] www.umtsworld.com/technology/wcdma.htm

[9] www.umts-forum.org o www.umtsforum.net

[10] www.gsmworld.com

[11] http://en.wikipedia.org/wiki/Perceptron

[12] www.3gpp.org

[13] Asignatura *Teoría de la Comunicación*, Universidad Carlos III de Madrid

[14] V. Kecman, M. Vogt, T. Huang, *On the Equality of Kernel AdaTron and Sequential Minimal Optimization in Classification and Regression Tasks and Alike Algorithms for Kernel Machines*, D-side Publications 2003

[15] Asignatura *Sistemas de Telecomunicación*, Universidad Carlos III de Madrid

[16] http://es.wikipedia.org/wiki/Relaci%C3%B3n_se%C3%B1al/ruido

[17] http://es.wikipedia.org/wiki/Wideband_Code_Division_Multiple_Access

[18] http://en.wikipedia.org/wiki/Internet_Protocol

[19] F. G. Stremler, *Introducción a los sistemas de comunicación*. Editorial Addison-Wesley Iberoamericana, 1993

[20] http://en.wikipedia.org/wiki/Base_station_subsystem

[21] http://es.wikipedia.org/wiki/Servicio_general_de_paquetes_v%C3%ADa_radio

[22] A. Bruce Carlson, Paul B. Crilly, Janet C. Rutledge, *Communication Systems: An introduction to signals and noise in electrical communication*. Editorial McGraw-Hill, 1986

[23] http://en.wikipedia.org/wiki/Hybrid_automatic_repeat_request

[24] http://en.wikipedia.org/wiki/High-Speed_Downlink_Packet_Access para conocer más técnicas que posibilitan el aumento de velocidad.

[25] http://en.wikipedia.org/wiki/Orthogonal_frequency-division_multiplexing

[26] http://en.wikipedia.org/wiki/MIMO

[27] La respuesta en frecuencia del filtrado Doppler se puede encontrar en la asignatura *Laboratorio de Comunicaciones Móviles*, Universidad Carlos III de Madrid

[28] Asignatura *Comunicaciones Digitales*, Universidad Carlos III de Madrid.

[29] John G. Proakis, Dimitris G. Manolakis, *Digital Signal Processing. Principles, Algorithms and Applications*. Editorial Prentice may, 1996

[30] Asignatura *Comunicaciones Móviles*, Universidad de Sevilla

[31] Biografía de Vladimir Vapnik en www.clrc.rhul.ac.uk/people/vlad/

[32] Más información en www.research.att.com/

[33] Se emplea el término ½ para facilitar la resolución del problema de optimización, de otra forma, habría que minimizar ||w|| sujeto a las mismas condiciones que en la ecuación (3.3.4).

[34] K. Crammer, J. Kandola, Y. Singer, *Online Classification on a Budget*. Advances in Neural Information Processing Systems 16 MIT Press, 2004

Algoritmos de Software

http://www.kyb.mpg.de/bs/people/spider/