Universidad de la República
Facultad de Ingeniería

Proyecto DeCA

PROYECTO DE FIN DE CARRERA

DETECCIÓN DE CONSUMOS ANOMALOS

Integrantes del grupo

Federico Decia
Matías Di Martino
Juan I. Molinelli

DeCA

Detección de Consumos Anómalos

Tutor: Ing. Alicia Fernandez

Montevideo - Uruguay
8 de mayo de 2011

Agradecimientos

Queremos agradecer en primer lugar a nuestras familias, por habernos brindado el apoyo incondicional con el que nos han acompañado a lo largo de la carrera y que sin dudas hicieron posible se realizara este y otros proyectos.

En segundo lugar, agradecemos a nuestra tutora por habernos guiado a lo largo de todo el proyecto, y por compartir toda su experiencia y conocimientos que nos ayudaron a ir por buen camino y tomar las decisiones adecuadas.

Este trabajo no hubiera sido posible sin la colaboración del personal de la división de detección de fraudes de UTE, agradecemos a cada uno de ellos por la asistencia brindada y los conocimientos que compartieron con nosotros. En particular queremos destacar la importante colaboración brindada por Juan Pablo Kosut.

No podemos dejar de agradecer a colegas y amigos como, Andres Alcarraz y Rodrigo Alonso entre otros por su ayuda.

Por último agradecemos a los docentes del curso de ”Int. al reconocimiento de Patrones”, entre ellos Pablo Cancela, Martín Rocamora y especialmente a Pablo Muse, quienes nos han brindado apoyo a lo largo de todo el proyecto y nos proporcionaron correcciones y sugerencias extremadamente útiles.

Resumen

En el Uruguay, UTE trabaja para detectar posibles clientes fraudulentos, sin embargo el gran número de clientes y la gran variedad de fraudes que se pueden cometer hacen de esta un tarea de gran complejidad. En este trabajo se estudia y aplica la teoría de reconocimiento de patrones a la detección de consumos anómalos (con sospechas de ser fraudulento). Los fraudes cometidos por clientes consumidores de energía eléctrica producen cuantiosas perdidas a las empresas distribuidoras. DeCA pretende aportar una herramienta que permita a los trabajadores de UTE detectar con mayor facilidad y eficiencia aquellos consumidores cuyas curvas de consumo presentan anomalías que justifican realizar una inspección. Para esto, se utilizaran clasificadores como One Class SVM, CS-SVM, OPF y el árbol C4.5 estabilizado con Adaboost, también se proponen estrategias novedosas para combinar estos clasificadores, y se utilizan medidas de performance como el F-value el Recall y Precision, teniendo especial consideración del problema de desbalance de clases. Luego, se presentan resultados obtenidos usando un conjunto de 1504 clientes proporcionados por UTE y se evalúa la performance en campo *mediante la realización de inspecciones. También se realizó una interfaz de usuario que permite utilizar todas las herramientas implementadas de manera sencilla y realizar pruebas con distintas base y estrategias.*

Palabras clave: *reconocimiento de patrones, UTE, consumos, detección de fraude, clasificadores, One Class SVM, CS-SVM, OPF, C4.5, Adaboost, F-value, Accuracy, Desbalance de clases.*

Índice general

Parte I

Descripción del Problema

CAPÍTULO 1

Introducción

1.1. Descripción del Problema

El uso irregular o fraudulento de la energía eléctrica representa un problema de gran magnitud provocando cuantiosas pérdidas a las empresas distribuidoras de muchos países o regiones. En el caso de Montevideo los balances de energía arrojan valores elevados de pérdidas totales. Este proyecto se enfoca en las pérdidas por fraude por lo que debemos separar de las pérdidas totales, las pérdidas técnicas en la red de distribución y las pérdidas asociadas a las zonas carenciadas. Las primeras por tener un origen ajeno al objeto de análisis, y las últimas por tratarse de pérdidas ya plenamente identificadas.

Detectar las pérdidas por fraude, es una tarea difícil fundamentalmente por dos razones. En primer lugar el gran número de clientes y en segundo lugar la gran variedad de fraudes y modos de alterar las medidas de consumo.

Solo en Montevideo existe cerca de 500.000 clientes, inspeccionar de manera visual cada curva de consumo requeriría una cantidad muy grande de recursos. Por esta razón, actualmente solo se observan algunos consumos de aquellos clientes con mayor potencia contratada o clientes comerciales que representan una porción menor del total de clientes. Además realizar inspecciones de manera visual, demanda una gran cantidad de tiempo y se vuelve un trabajo muy monótono, como el índice de infractores es bajo (menor al 5 %) es natural que luego de horas de observar consumos la atención baje y la eficiencia del reconocimiento decaiga.

Por otro lado la gran cantidad de tipos de fraude complica aun más el problema de la detección visual por medio del estudio de la curva de consumo, ya que existen muchas posibilidades que pueden estar encubriendo actividades fraudulentas. A modo de ejemplo, aquellos consumidores que se *conectan* y *desconectan* al contador regulan la cantidad de kW que consumen en el mes, y las curvas de consumo de estos clientes presentan varianzas muy pequeñas (en comparación con la dispersión normal), otros tipos

de fraude como modificar el contador introduciendo una pequeña resistencia mecánica, haciendo que el valor medio del consumo baje, trasladando la curva de consumo pero sin modificar su forma. Otros clientes infractores modifican el esquema de conexión o ponen a tierra alguna fase introduciendo cambios en las curvas de consumo que venían presentando. En algunos casos el fraude se manifiesta por medio de un *cambio en la estacionalidad* del consumo, por ejemplo, autoservicios con muchas heladeras consumen energía eléctrica en función de la temperatura. Estos consumos presentan grandes correlaciones con la temperatura y una periodicidad muy marcada, un cambio en cualquiera de estos aspectos puede delatar actividades fraudulentas o alteraciones en los contadores.

Si repasamos las formas en que un consumo puede manifestarse ante una actividad fraudulenta y tenemos en cuenta una gran cantidad de otras maneras que no se mencionaron o aun no se conocen, podemos entender la complejidad que implica detectar de manera visual consumos anómalos que ameritan ser inspeccionados. Recordemos que el problema no es únicamente la atención que se requiere para detectar cualquiera de los aspectos antes mencionado, sino la cantidad de tiempo que es necesario mantener dicha atención al visualizar uno y otro consumo durante horas. Aun realizando dicha actividad, demandaría meses de tiempo observar todos los clientes de Montevideo, sin mencionar que el control debe hacerse de manera periódica lo cual implicaría un trabajo permanente.

1.2. Objetivos del Proyecto

El objetivo principal del proyecto es la generación de una herramienta que permita, en base a un conjunto de consumos etiquetados, detectar nuevos consumos anómalos.

En segundo lugar, se pretende estudiar las herramientas existentes, adaptándolas al problema concreto y proponiendo nuevas estrategias en función de las necesidades que surjan del proyecto.

1.2.1. Comentarios generales de los objetivos

Cabe resaltar que no se pretende desarrollar una herramienta que automatice la detección de clientes fraudulentos o medidores dañados. Para determinar posibles hechos fraudulentos es necesario tomar en consideración una gran cantidad de factores además de observar la curva de consumos. A modo de ejemplo, se debe tener en cuenta la cantidad de metros cuadrados del local (en comparación con la potencia solicitada), si el cliente posee antecedentes de fraude, el barrio en el que se encuentra el cliente, entre otros factores.

A pesar de lo anterior, este proyecto brinda un herramienta que permite al personal de UTE aprovechar los recursos humanos. En lugar de estudiar a todos los clientes (hecho que en el la práctica es imposible), los algoritmos que se desarrollan en este proyecto clasifican solo una pequeña porción del conjunto de clientes con la categoría de consumos anómalos. La idea general es que dentro de esta porción de clientes

seleccionados se encuentren aquellos con mayor probabilidad de presentar fraude. De este modo se podrá abarcar el estudio de un número muy grande de clientes aún con recursos humanos limitados y capacidades de inspección moderadas.

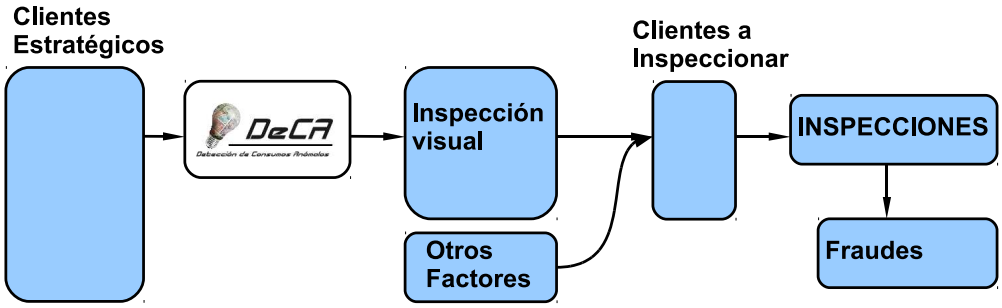


Figura 1.1: Detección de fraude agregando la detección automática de consumos anómalos al proceso.

La estrategia que se plantea será mucho más eficiente que seleccionar clientes a inspeccionar de manera aleatoria. Además, como se tiene por efecto separar aquellos consumos que por alguna razón presentan diferencias (o anomalías) respecto al conjunto que se utiliza para entrenar los algoritmos, se detectarán, además de fraudes posibles, fallas en la facturación o en los medidores (que no necesariamente estén asociadas a un hecho delictivo por parte del cliente).

Existen también, objetivos de carácter *académico*. Si se consulta la bibliografía más reciente y las publicaciones relacionadas con la detección de fraude en energía eléctrica [29][30], se puede observar que el tema no ha sido muy estudiado a nivel mundial. Si bien se pueden encontrar un número pequeño de publicaciones que abordan el tema, fundamentalmente en Brasil y en Malasia, en general son estudios aislados sin mecanismos de detección incorporados a las empresas distribuidoras de energía eléctrica. Por otro lado, y en un plano mucho más general, la detección de fraude es un objeto de estudio que concierne a varios rubros como el bancario y los servicios de telecomunicaciones entre otros, en todos estos casos la detección de fraude tiene aspectos en común. En la mayoría de las aplicaciones se deben enfrentar retos como por ejemplo, la baja proporción de muestras *fraudulentas* comparado con las *normales*, la gran variedad en los tipos de fraude y las características a extraer de los datos disponibles entre otros. Por estos motivos en este trabajo se plantean también como objetivos, el estudio de las distintas técnicas que existen para abordar de manera formal estos problemas y se propone una solución que en función de los estudios realizados, se adapta mejor al problema en concreto que se pretende abordar. Si bien la solución esta orientada a brindar una herramienta que ayude en el proceso de detección de fraude en energía eléctrica muchas de las estrategias aquí propuestas se pueden adaptar a otros problemas sin mayores cambios.

1.3. Descripción del documento.

En este documento procuraremos transmitir y ordenar todo el trabajo realizado en el marco del proyecto de fin de carrera “DeCA - Detección de Consumos Anómalos”. Dividimos el trabajo en 3 partes, en la primera, comenzaremos describiendo el problema que se pretende abordar, luego mostraremos las técnicas que se utilizaran para encontrar la solución al problema y daremos el esquema de la solución propuesta sin entrar en detalles técnicos o teóricos.

En la segunda parte, daremos marco formal a las herramientas utilizadas. En el capitulo 5 introducimos un marco teórico a los métodos de selección de características utilizados, siguiendo en el capítulo 6 con una descripción teórica de los clasificadores utilizados. Luego en el capítulo 7 mostraremos las distintas técnicas disponibles de combinación de clasificadores enfatizando aquellas que serán utilizadas en la solución propuesta.

Por último en la tercera parte del documento, mostramos los detalles de la implementación de la solución al problema, describiendo las realizaciones puntuales de los algoritmos que fueron elegidos, las características propuestas (capítulo 9), los métodos de selección utilizados (capítulo 10), los clasificadores que se implementaron con sus respectivas estrategias para el entrenamiento y determinación de parámetros (capítulo 11) y por último las distintas opciones de combinación de clasificadores (capítulo 12).

El documento finaliza con la presentación de los resultados y las conclusiones obtenidas en el capítulo 14 donde se detallan también algunas líneas en las que se puede continuar trabajando y otros aspectos que quedan planteados para futuros abordajes de esta problemática (sección 14.2).

CAPÍTULO 2

Abordaje

2.1. Antecedentes

Se comenzó a explorar el campo del reconocimiento de patrones y su utilidad para solucionar problemas de detección de consumos anómalos en el año 2008, en el trabajo desarrollado por Juan Pablo Kosut y Diego Alcetegaray en el marco del proyecto de final del curso "Int. al Reconocimietno de Patrones" dictado por el IIE, denominado **ONE CLASS SVM para la detección de fraudes en el uso de energía eléctrica** [10]. En el mencionado trabajo se atacó el problema a partir de una base de datos de aproximadamente 650 suministros pertenecientes a los rubros almacenes y autoservicios de la base de datos de UTE, aplicando el método **One-Class SVM** con la idea de que existe *una sola clase*, los suministros normales, mientras que los suministros anómalos son considerados *outliers*.

En una primera etapa se comenzó etiquetando estos suministros de forma manual por los propios técnicos de UTE para luego poder medir la performance del clasificador desarrollado. Luego, se eligieron características como *desviación estándar del consumo sobre la media*, *valor de la pendiente al aproximar el consumo por una recta* y *los primeros 3 coeficientes de fourier*, y a partir de ellas se generó un primer modelo de clasificación evaluando su capacidad para discernir entre suministros anómalos y normales. Con este primer modelo de clasificación se compararon las etiquetas obtenidas con el clasificador y las etiquetas manuales concluyendo que se habían cometido ciertos errores en el etiquetado manual por lo que se decidió re-etiquetar las muestras donde se habían cometido los errores y volver a repetir el algoritmo de entrenamiento y creación de un nuevo modelo. Los resultados finales obtenidos para esta base fueron muy alentadores, con errores de clasificación del $0\,\%$ para los consumos anómalos y menores que el $13\ \%$ para consumos normales.

Sin embargo, esta primera aproximación al problema es muy acotada ya que la base de datos utilizada es muy pequeña y solo representa a un pequeño tipo de

suministros además de existir un gran overfitting al haber realizado el re-etiquetado de alguna muestras.

Por su parte Jawad Nagi et al. [30] atacaron el problema de detección de perdidas no-técnicas para una compañía malaya. Propusieron también una aproximación a partir de maquinas de vector soporte pero con una modificación, partiendo de la base de la existencia de dos clases, anómalos y normales, por lo que el método utilizado fue **C-SVM**.

2.2. Solución Propuesta

Una vez establecido el problema que se pretende abordar, describiremos de que manera podemos obtener una solución a los objetivos planteados de manera efectiva. La idea de este proyecto es brindar herramientas mediante el uso de técnicas de reconocimiento de patrones que permitan auxiliar al personal de UTE en la detección visual de consumos anómalos que deben ser inspeccionados.

El diagrama de bloques con un esquema de la solución propuesta se muestra en la figura 2.1, a continuación se introducen las características principales de cada bloque. En la última parte de este documento se entrará en detalle describiendo con mayor profundidad los procedimientos que se realizan dentro de cada uno de estos bloques.

Como cualquier problema de reconocimiento de patrones, el primer paso consiste en realizar un preprocesamiento de los datos. Esta etapa consiste en preparar las muestras para eliminar *impurezas* que puedan introducir ruido al sistema y alterar el desempeño del mismo. Los detalles del preprocesamiento realizado se presentan en el capítulo 8. En la etapa de preprocesamiento se prepara los datos para las etapas posteriores atendiendo particularmente los siguientes aspectos:

- Aquellos consumos que presentan una fecha de detección de irregularidad, son cortados para que los mismos tengan como fecha final la fecha en la que se constato la irregularidad. Luego se les asigna la etiqueta de consumo *anómalo* (o sospechoso). Esta etiqueta será utilizada por los algoritmos para determinar los parámetros óptimos de los clasificadores y fijar las fronteras entre las clases.

- Aquellos consumos que presentan muchos periodos de 0 (muchos meses sin consumo alguno de energía eléctrica) son descartados. En general estos consumos presentan cambios de firma o fincas abandonadas que no son el objeto de este trabajo y podrían introducir ruido en la etapa de entrenamiento.

- Las medidas de consumo mensual pueden estar determinadas de distintas maneras. En algunos casos la lectura corresponde a la lectura tomada por el personal de UTE, en otros casos la lectura puede ser *estimada*, por ejemplo, cuando se carece de lectura para un determinado mes o cuando es provista por los propios clientes. En cualquiera de los dos últimos casos se puede generar un desajuste entre lo que marca el contador de los clientes y lo que lleva como registro de consumo la base de datos de UTE. Cuando este desajuste se corrige (mediante una lectura del personal

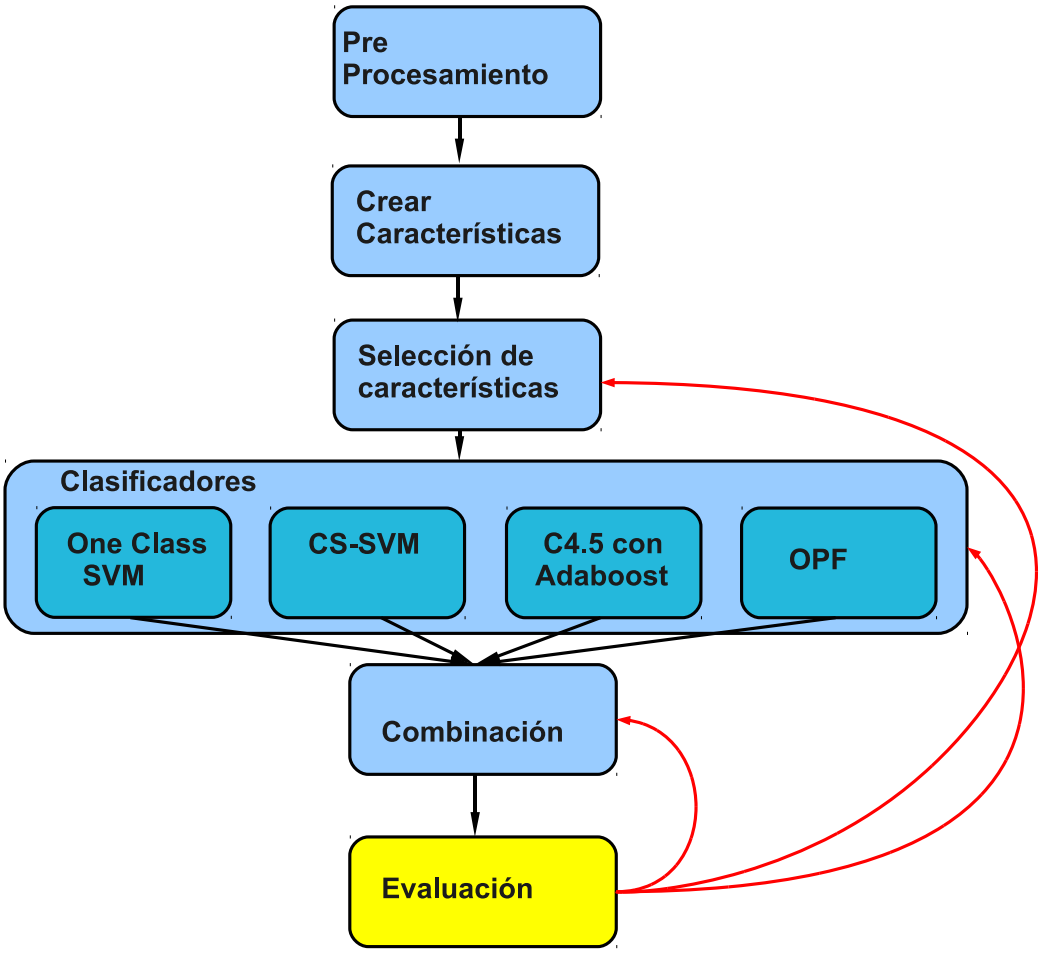


Figura 2.1: Diagrama de bloques

fiscalizador de UTE) se producen saltos en la curva de consumo. Por esta razón es necesario durante el preprocesamiento eliminar algunos *picos* que surgen de desajustes por la forma en que se obtienen las medidas de consumos pero que no representan un pico de consumo mensual real.

- Los consumos también son normalizados para trabajar con una base de datos homogénea desde el punto de vista de la *escala* en la que se observan los consumos.

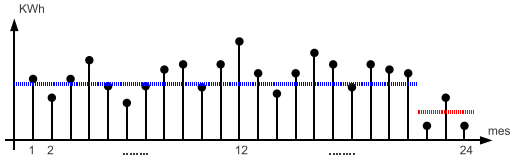
- Otro aspecto importante a realizar en el preprocesamiento se refiere a preparar el *formato* de los datos. De este modo todos los procesos que reciben datos ya conocen el formato (largo de los consumos, formato de las matrices, entre otros) en el que se manejan. De esta forma garantizamos que dado el caso que se quiera agregar bases de datos que provienen de fuentes no uniformes, solo debemos realizar adaptaciones y modificaciones en la entrada del proceso pero no en todos los bloques subsiguientes.

Una vez que preparamos los datos teniendo en cuenta los aspectos puntualizados anteriormente, podemos pasar a representar los consumos en un *espacio adecuado*. Antes

de estudiar qué algoritmos se pueden utilizar para separar aquellos consumos *normales* de los *anómalos* debemos representar a las muestras mediante un vector de características (que generalmente pertenece a $\mathbb{R}^n$, en algunos contextos se pueden utilizar características de otro tipo como pueden ser: binarias o "colores", etc). Las características que se plantearon en este trabajo fueron las siguientes:

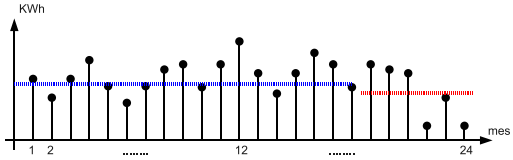
1. Cociente entre el valor medio en los últimos 3 meses y el pasado

$$car1 = \frac{mean(C[n-3:n])}{mean(C([1:n-4]))} \tag{2.1}$$



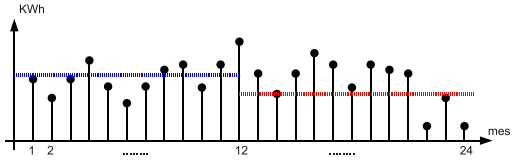
2. Cociente entre el valor medio en los últimos 6 meses y el pasado

$$car2 = \frac{mean(C[n-6:n])}{mean(C([1:n-7]))} \tag{2.2}$$



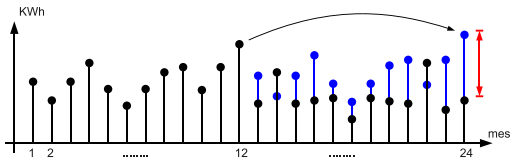
3. Cociente entre el valor medio en los últimos 12 meses y el pasado

$$car3 = \frac{mean(C[n-12:n])}{mean(C([1:n-13]))} \tag{2.3}$$



4. Norma de la diferencia entre el consumo esperado y el consumo real

$$car4 = \sqrt{\sum_{i=n-11}^{i=n} \left(C(i) - \alpha C(i-12)\right)^2} \tag{2.4}$$



5. Diferencia en los espectros

$$car5 = \|FFT\{C_{actual}\}(1) - FFT\{C_{medio}\}(1)\| \tag{2.5}$$
$$car6 = \|FFT\{C_{actual}\}(2) - FFT\{C_{medio}\}(2)\| \tag{2.6}$$
$$car7 = \|FFT\{C_{actual}\}(3) - FFT\{C_{medio}\}(3)\| \tag{2.7}$$

6. Diferencia en los coeficientes wavelet

$$car8 = |cA2_u(1)| - |cA2_p(1)| \quad (2.8)$$
$$car9 = |cA2_u(2)| - |cA2_p(2)| \quad (2.9)$$
$$car10 = |cA2_u(3)| - |cA2_p(3)| \quad (2.10)$$
$$car11 = |cA2_u(4)| - |cA2_p(4)| \quad (2.11)$$
$$car12 = |cA2_u(5)| - |cA2_p(5)| \quad (2.12)$$
$$car13 = \|cD2_u - cD2_p\| \quad (2.13)$$
$$car14 = \|cD1_u - cD1_p\| \quad (2.14)$$

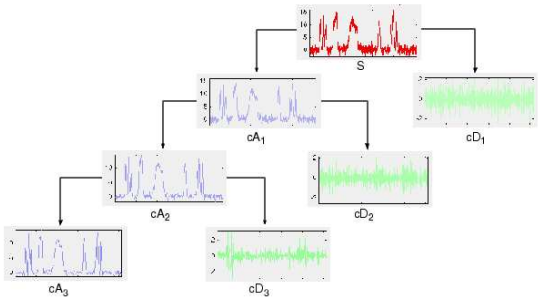
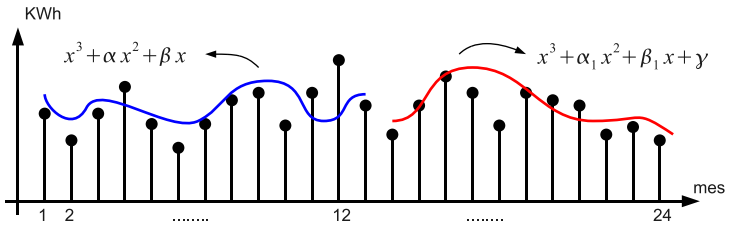


Figura 2.2: Descomposición multinivel. Imagen extraída del manual de usuario del ”Wavelet Toolbox” de Matlab.

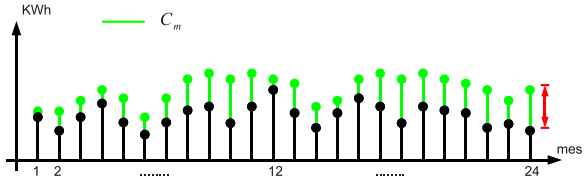
7. Diferencia en el ajuste de un polinomio de grado n

$$car\{15, 16, 17, 18, 19\} = polyfit(C_{ao\ n}) - \frac{1}{n-1}\sum_{i=1}^{n-1} polyfit(C_{ao\ i}) \quad (2.15)$$



8. Distancia al consumo medio

$$car20 = \left\|\vec{C} - \vec{C}_m\right\| \quad (2.16)$$



9. Comparación de la varianza del consumo

$$car21 = \frac{var\,(C_N)}{var\left(\frac{1}{N-1}\sum_{i=1}^{N-1} C_i\right)} \quad (2.17)$$

10. Varianza del consumo

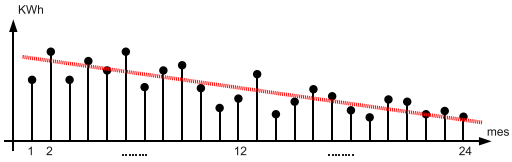
$$car22 = \frac{var\,(C_N)}{var\,(C_m)} \quad (2.18)$$

11. Coeficientes de Fourier

$$car\{23, 24, 25, 26, 27\} = \left\| FFT(C)_{\{1,2,3,4,5\}} \right\| \tag{2.19}$$

12. Pendiente de la recta que ajusta al consumo

$$car28 = polyfit(C, 1) \tag{2.20}$$



El detalle de cada una de las características antes listadas se encuentra en el capítulo 9, donde se entrara en mayor profundidad en cada una de ellas.

El conjunto de características anterior, fue resultado de las distintas reuniones que se realizaron con el personal encargado de la detección de fraude en UTE. Las mismas intentan plasmar aspectos que los técnicos tienen en cuenta a la hora de evaluar si una curva de consumos presenta un comportamiento sospechoso o no. Por otro lado se contemplaron ciertas características que no surgen del intercambio con los expertos en el área. Estás fueron propuestas en base a publicaciones recientes en el área de detección de consumos anómalos y en trabajos previos realizados por el propio personal de UTE.

Si bien la lista anterior es el conjunto de características finales que fueron propuestas, en este trabajo se utilizarán también otros dos conjuntos de características, el propuesto en [10] y el conjunto de consumos. Es decir, se contempla la posibilidad de que el vector de consumos sea utilizado directamente, tomando como característica el valor mensual de cada consumo. La idea de utilizar como características los consumos tiene básicamente dos sustentos, por un lado es el esquema utilizado en algunas de la publicaciones más recientes [30], y por otro comparar el resultado obtenido utilizando los vectores originales como características con el resultado obtenido utilizando las características propuestas. De esta manera podremos responder a la siguiente pregunta: ¿Vale la pena representar a los consumos mediante *nuevas* características o es una complicación adicional al sistema que no se justifica?

Una vez establecido el conjunto de características, se implementa una etapa de selección. Trabajar con un número muy elevado de características demanda mayores cantidades de tiempo y puede degradar la performance de algunos clasificadores. En muchas ocasiones agregar características que no discriminan entre las clases tiende a empeorar el desempeño de los clasificadores, además en general se requiere que el número de muestras de entrenamiento aumente de manera exponencial al aumentar la dimensión (número de características) del problema, esto se conoce como “Maldición de la dimensión”[20]. Por este motivo se implementaron métodos de selección de características que permiten obtener del conjunto original aquel subconjunto que mejor discrimina entre las clases.

A la hora de definir los métodos de selección de características, es necesarios especificar:

1. Los métodos de búsqueda.

2. Los métodos de evaluación.

El primer punto, consiste en especificar el modo en que se pretende encontrar el subconjunto *óptimo* de características y, el segundo consiste en establecer los criterios que determinan que subconjunto es mejor que otro.

Dentro de los métodos de búsqueda se utilizo en algunos casos la búsqueda exhaustiva y en la mayoría de los casos el método de *bestfirts*. Si bien el segundo no garantiza obtener el subconjunto óptimo, es mucho menos costoso desde el punto de vista computacional y ha arrojado muy buenos resultados.

Si tomamos en consideración los métodos de evaluación, tenemos básicamente dos enfoques, por un lado se pueden utilizar algoritmos de tipo *filter*, que procurar eliminar redundancia entre las características y tratar de captar aquellas que tienen mayor correlación con las clases. Estos algoritmos no tienen en cuenta qué clasificador se va a utilizar y son independientes de los mismos. Por el otro lado, existen los algoritmos llamados *wrapper* que, buscan maximizar la performance para un clasificador dado.

Los métodos utilizados en este trabajo se listan a continuación:

- *Wrapper* con el clasificador **C-SVM** buscando maximizar tanto el *accuracy*[a] como el F-value[b]. Teniendo en cuenta que **C-SVM** es uno de los clasificadores utilizados, resulta lógico aplicarle este método de evaluación. Se busca maximizar el *accuracy* porque es una medida estándar y es la más intuitiva y utilizada, sin embargo en problemas de clases desbalanceadas, como éste, donde además el costo de clasificar mal una clase y otra es diferente resulta más razonable maximizar el F-value[17].

- *Wrapper* con el clasificador **C4.5** buscando maximizar tanto el *accuracy* como el F-value. Aquí, aplican las mismas razones que las explicadas anteriormente.

- *Wrapper* sensible al costo con el clasificador **C4.5** buscando maximizar el *accuracy*. Se aplico esta variación al método para analizar la incidencia que podía tener.

- *Wrapper* con el clasificador **Vecino mas cercano**. Se decidió probar este método de evaluación porque los algoritmos de clasificación de **Vecino más cercano** y **OPF** tienen puntos en común.

- CfsSubsetEval como método *filter*, ya que vimos que se pueden obtener buenos resultados, además de permitir realizar una búsqueda exhaustiva como método de selección.

[a] El accuracy es una medida estándar en los problemas de reconocimiento de patrones y se calcula como el número de instancias correctamente clasificadas sobre el número total de muestras.

[b] En problemas donde las clases están desbalanceadas, es necesario considerar otras medidas de performance además del accuracy. EL F-value es un indicador que toma en cuenta tanto el número de anómalos bien clasificados (Recall) como la densidad de anómalos dentro de los consumos clasificados como anómalos (Precision), $F_{value} = \frac{Recall \times Precision}{Recall + Precision}$. Cuando se describan las técnicas para el abordaje de problemas con clases desbalanceadas se trataran con profundidad todos estos aspectos.

Los detalles teóricos de los métodos de selección utilizados se introducen en el capítulo 5 mientras que los detalles de las implementaciones puntuales serán abarcados en el capítulo 10.

Una vez que se determinó un conjunto de características óptimo, eliminando información redundante y características que no aportan en la discriminación de las clases, es momento de entrenar y determinar los parámetros de cada uno de los clasificadores que se van a utilizar. Antes de comenzar a describir qué clasificadores vamos a utilizar, corresponde aclarar que el conjunto de características seleccionado puede depender de cada clasificador. Es decir, para cada uno de los clasificadores que se tomó en consideración, se determina qué subconjunto de características se adapta mejor y arroja mejores resultados. Luego, para cada clasificador tenemos determinadas las características que se utilizarán.

Hecha la apreciación anterior, veamos qué clasificadores serán los responsables de clasificar a cada muestra como *anómala* o *normal*. La elección de los clasificadores a la hora de abordar un problema de reconocimiento de patrones es una tarea crucial que condiciona fuertemente el desempeño y la robustez que tendrá el sistema. Por otro lado no existen reglas claras que dicten qué clasificador se debe utilizar en cada caso, en general las soluciones dependen de cada problema en concreto. Por esta razón se prefirió utilizar un *conjunto de clasificadores* en lugar de proponer un único clasificador como proponen algunos trabajos[30][29][10].

Los clasificadores que se utilizarán son:

- One Class SVM
- CS - SVM (cost sensitive C-SVM)
- OPF
- C4.5 con Adaboost

Los clasificadores basados en SVM (tanto One Class SVM y CS-SVM) procuran encontrar un hiperplano que separa a las muestras de una y otra clase. OPF en cambio, es un algoritmo que utiliza una distancia para asignar a cada muestra aquella clase del prototipo *más cercano*. Por último se utilizara un árbol (C4.5) que consiste en realizar preguntas simples e ir separando a las muestras de una clase y otra.

Los detalles teóricos de cada uno de los algoritmos utilizados se presentan en el capítulo 6. Por otro lado discutimos la implementación de cada uno de los clasificadores y las consideraciones tenidas en cuenta a la hora de la puesta a punto de cada uno de los métodos en el capítulo 11.

Con los clasificadores bien establecidos y los detalles de cada uno de los algoritmos propuestos ultimados, debemos determinar de qué modo se *combina* la información obtenida a partir de cada uno de ellos. Combinar las salidas de cada uno de los clasificadores de manera adecuada es determinante para que el desempeño del conjunto supere al mejor desempeño individual. De lo contrario no se justifica una etapa

de combinación de clasificadores (es más eficiente en ese caso considerar al mejor del conjunto sin mayores complicaciones).

Antes de plantear cualquier estrategia de combinación se debe tener en cuenta la naturaleza de cada uno de los clasificadores que se desea combinar respondiendo las siguientes interrogantes: ¿Qué tipo de información arroja cada clasificador? ¿Disponemos únicamente de la etiqueta o se tiene información asociada a la confianza de la clasificación? ¿El desempeño del clasificador es uniforme en todo el espacio de las muestras? ¿Cómo es el desempeño de cada uno de los clasificadores en comparación? En la medida en que se resuelvan estas interrogantes, y se tengan ajustados cada uno de los clasificadores *base*, se pueden plantear distintas estrategias de combinación que, en base a la utilización de la información arrojada por cada uno de los clasificadores construya un método de decisión más preciso que cada uno de los métodos individualmente.

Los algoritmos de combinación de clasificadores considerados se enumeran a continuación. Muchos de estos métodos son diseñados en este trabajo.

En cualquiera de los esquemas de clasificación propuestos, se definen g_p y g_n para cada una de las muestras, éstas se calculan en función de las etiquetas asignadas por cada clasificador. Luego, se asignara a un consumo x la etiqueta anómalo si $g_p(x) > g_n(x)$ y normal en el caso contrario.

- Todo Accuracy.
  Este método es una pequeña modificación del método conocido como *voto por mayoría ponderada*. El peso que tiene cada uno de los clasificadores base se calcula en función de su Accuracy. De este modo definimos :

$$\begin{aligned} g_p(x) &= d_p^1(x)\, log\left(\frac{Acc_1}{1-Acc_1}\right) + d_p^2(x)\, log\left(\frac{Acc_2}{1-Acc_2}\right) \\ &\quad + d_p^3(x)\, log\left(\frac{Acc_3}{1-Acc_3}\right) + d_p^4(x)\, log\left(\frac{Acc_4}{1-Acc_4}\right) \end{aligned}$$

$$\begin{aligned} g_n(x) &= d_n^1(x)\, log\left(\frac{Acc_1}{1-Acc_1}\right) + d_n^2(x)\, log\left(\frac{Acc_2}{1-Acc_2}\right) \\ &\quad + d_n^3(x)\, log\left(\frac{Acc_3}{1-Acc_3}\right) + d_n(x)^4\, log\left(\frac{Acc_4}{1-Acc_4}\right) \end{aligned}$$

- Accuracy.
  Se modifica el esquema anterior considerando en lugar de la Accuracy global de cada uno de los clasificadores para determinar los pesos, la confianza individual que se tiene sobre cada una de las muestras (para aquellos clasificadores para los cuales la información está disponible).

$$\begin{aligned} g_p(x) &= d_p^1(x)\, log\left(\frac{Acc_1}{1-Acc_1}\right) + d_p^2(x)\, log\left(\frac{P_p^2(x)}{1-P_p^2(x)}\right) \\ &\quad + d_p^3(x)\, log\left(\frac{P_p^3(x)}{1-P_p^3(x)}\right) + d_p^4(x)\, log\left(\frac{Acc_4}{1-Acc_4}\right) \end{aligned}$$

$$g_n(x) = d_n^1(x)\, log\left(\frac{Acc_1}{1-Acc_1}\right) + d_n^2(x)\, log\left(\frac{P_n^2(x)}{1-P_n^2(x)}\right) + d_n^3(x)\, log\left(\frac{P_n^3(x)}{1-P_n^3(x)}\right) + d_n(x)^4\, log\left(\frac{Acc_4}{1-Acc_4}\right)$$

- Todo Diferencial.
  En lugar de establecer el peso de cada uno de los clasificadores en función de su Accuracy, se toma en cuenta el Acc_+[c] para establecer la confianza sobre las muestras etiquetadas como normales (o negativas) y el $Acc\_$[d] para aquellas etiquetadas como anómalas (o positivas). Este esquema fue más eficiente para tratar con este problema en el cual las clases están desbalanceadas. Se define en este caso:

$$g_p(x) = d_p^1(x)\, Acc_n^1 + d_p^2(x)\, Acc_n^2 + d_p^3(x)\, Acc_n^3 + d_p^4(x)\, Acc_n^4 \tag{2.21}$$

$$g_n(x) = d_n^1(x)\, Acc_p^1 + d_n^2(x)\, Acc_p^2 + d_n^3(x)\, Acc_p^3 + d_n^4(x)\, Acc_p^4 \tag{2.22}$$

- Diferencial.
  Para los clasificadores para los cuales se cuenta con la confianza de la etiqueta asociada a cada muestra, ésta se utiliza como peso del clasificador. Para el resto se utiliza, Acc_+ para las muestras etiquetadas como normales y $Acc\_$ para aquellas etiquetadas como anómalas. Definimos entonces:

$$g_p(x) = d_p^1(x)\, Acc_n^1 + d_p^2(x)\, P_p^2(x) + d_p^3(x)\, P_p^3(x) + d_p^4(x)\, Acc_n^4$$

$$g_n(x) = d_n^1(x)\, Acc_p^1 + d_n^2(x)\, P_n^2(x) + d_n^3(x)\, P_n^3(x) + d_n^4(x)\, Acc_p^4$$

- Iterativo.
  Es similar al esquema de voto por mayoría ponderara. La diferencia radica en que en lugar de calcular los pesos de cada clasificador como una función de la probabilidad de acierto, los pesos se obtienen mediante búsqueda exhaustiva maximizando el F_{value} de la clasificación sobre una base de entrenamiento.

$$g_p(x) = d_p^1(x)\, b_1 + d_p^2(x)\, b_2 + d_p^3(x)\, b_3 + d_p^4(x)\, b_4$$
$$g_n(x) = d_n^1(x)\, b_1 + d_n^2(x)\, b_2 + d_n^3(x)\, b_3 + d_n^4(x)\, b_4$$

En el capítulo 7 se introducen los aspectos teóricos que serán utilizados en el capítulo 12 cuando se detallen las implementaciones de lo métodos de combinación antes mencionados.

Con los esquemas de combinación propuestos, culminamos con la descripción del *último* de los bloques que se ilustran en la figura 2.1. Si bien el diagrama de bloques mostrado en la figura establece un orden para los bloques, mostrando un sistema en cascada que parece estar bien definido y con orden de precedencia marcado, en la realidad el ajuste y diseño de cada bloque depende de los otros. En la figura 2.1 ilustramos este

[c] Acc_+ se calcula como el número de consumos anómalos correctamente clasificados sobre el total de consumos anómalos

[d] $Acc\_$ se calcula como el número de consumos normales correctamente clasificados sobre el total de consumos normales

hecho por medio de los lazos de realimentación que se dibujan en color rojo. Solamente cuando se tiene el sistema implementado de principio a fin, se puede comenzar a optimizar y diseñar correctamente cada uno de los bloques atendiendo a la performance final que tiene el esquema propuesto. Por esta razón, si bien en este capítulo se optó por describir cada bloque en orden, como si fueran independientes, se debe tener en cuenta que en la práctica el proceso fue bien distinto. A la hora de diseñar los distintos bloques, se buscó optimizar todo el mecanismo de clasificación. Cada vez que se realizaba algún cambio se alteraba todo el proceso y algunas de las decisiones tomadas antes de los cambios debían ser puestas a prueba en el nuevo contexto. Por esta razón, el proceso de diseño del método de clasificación *óptimo* es un proceso iterativo donde se van optimizando detalles que alteran otras configuraciones ya establecidas.

A pesar del carácter iterativo que presenta el proceso de diseño de un clasificador *óptimo*[e] como se menciona en el párrafo anterior, se tiene la convicción de que el esquema propuesto es lo suficientemente general como para adaptarse a las exigencias que se proponen y para cumplir con los objetivos propuestos. El conjunto de características, los métodos de selección, los clasificadores considerados y los algoritmos de combinación propuestos, son lo suficientemente amplios y contemplan todas las estrategias utilizadas en la literatura más reciente referida al tema. De este modo, el esquema final es un modelo robusto que permite atacar el problema de detección de consumos anómalos y brindar apoyo al personal de UTE a la hora de detectar fraude. Por otro lado, tanto los resultados teóricos como los resultados en campo que se presentarán, mostraron que la herramienta desarrollada es prometedora y se adapta al problema que pretende abordar.

[e]Por clasificador óptimo debe entenderse aquel que mejora cierto indicador (que se debe definir adecuadamente en función del problema) en este trabajo, el clasificador óptimo será aquel que arroje mejor F-value en la clasificación.

Parte II

Marco Teórico

CAPÍTULO 3

Aspectos generales del Reconocimiento de Patrones

3.1. ¿Qué es el Reconocimiento de Patrones?

Según Anil Jain [20], el reconocimiento de patrones es el estudio de cómo las máquinas pueden observando el ambiente aprender a distinguir **patrones** de interés de un fondo y realizar decisiones razonables sobre las categorías de los mismos. Por lo tanto comenzamos definiendo un patrón. Watanabe [44] define a un **patrón** como al opuesto al caos, una entidad definida vagamente a la cual se le puede dar un nombre. Por ejemplo, un patrón puede ser una huella digital, un rostro humano, o una letra.

Dado un patrón, el reconocimiento (también denominado clasificación) consiste en cualquiera de las siguientes dos tareas: 1) Clasificación supervisada, en donde el patrón de entrada es identificado como miembro de una clase (tipo) predeterminada, 2) Clasificación no-supervisada, en donde el patrón de entrada es asignado a una clase hasta ahora desconocida. El problema de reconocimiento de patrones se plantea entonces como una tarea de clasificación donde las clases están definidas por el diseñador del sistema (clasificación supervisada) o se deben aprender basándose en la similitud de los patrones (clasificación no-supervisada).

El interés en el área del reconocimiento de patrones ha crecido en el último tiempo debido a la aparición de nuevas aplicaciones (cuadro 3.1) y al aumento en el poder de procesamiento computacional que ha permitido procesar con mayor rapidez conjuntos de datos de grandes dimensiones y facilitado el uso de diversos métodos para el análisis y clasificación de datos. En muchas de estas nuevas aplicaciones se ha visto que no existe un único enfoque de clasificación óptimo si no que se deben combinar multiples métodos y enfoques. En consecuencia, es común en el reconocimiento de patrones combinar varias modalidades de detección y clasificadores.

El diseño de sistemas de reconocimiento de patrones implica 3 aspectos: 1)Adquisición de los datos y preprocesamiento, 2) representación de los datos y 3) toma

Dominio	Aplicación	Patrón entrada	Clases
Data mining	Búsqueda patrones significativos	Puntos en espacio multidimensional	Clusters compactos y bien separados
Clasificación de documentos	Búsqueda en internet	Documentos de texto	Categorías semánticas
Reconocimiento biométrico	Identificación personal	Cara, iris, huella digital	Control de acceso para usuarios autorizados
Reconocimiento de voz	Acceso a información sin operador	Señal de voz	Palabras habladas

Cuadro 3.1: Ejemplos de aplicaciones sobre Reconocimiento de Patrones

de decisiones. Por lo tanto, es necesario definir el/los método/s de detección, técnica de preprocesamiento, esquema de representación y el modelo para la toma de decisiones. Existe consenso en que un problema de reconocimiento *bien definido y limitado* (pequeñas variaciones intra-clase y grandes variaciones entre clases) llevarán a una representación de patrones compacta y a una estrategia simple para la toma de decisiones. En la mayoría de los sistemas de reconocimiento de patrones se busca *aprender* a partir de un conjunto de ejemplos, denominado base de entrenamiento [20]. Las 4 aproximaciones al reconocimiento de patrones más populares son:

1. Correspondencia de modelos (Template Matching)
2. Reconocimiento estadístico o geométrico de patrones
3. Reconocimiento sintáctico o estructural de patrones
4. Redes Neuronales

3.1.1. Template Matching: Correspondencia de Modelos

La comparación es una operación natural (o común) del reconocimiento de patrones utilizada para determinar la similitud entre dos entidades (puntos, curvas, etc.) del mismo tipo. En *template matching* se tiene un prototipo del patrón que se quiere reconocer por lo que se compara el patrón que se quiere clasificar con el prototipo disponible considerando todas las posturas (traslación y rotación) y cambios de escala posibles. La medida de similitud, en general la correlación, puede ser optimizada en base al conjunto de datos de entrenamiento.

3.1.2. Reconocimiento estadístico

En esta aproximación cada patrón es representado a partir de d características o medidas y es considerado como un punto en un espacio de dimensión d. El objetivo es elegir características tales que permitan que los vectores patrones de diferentes clases ocupen regiones compactas y disjuntas del espacio de características de dimensión d. La efectividad del espacio de características se determina en base a cuán bien se pueden separar patrones de clases diferentes. Dado un conjunto de patrones de entrenamiento de cada clase, el objetivo es establecer limites de decisión en el espacio de características

que separen patrones de clases diferentes. En el enfoque de clasificación estadística, es necesario conocer la distribución de probabilidad de los patrones de cada clase para determinar los limites de decisión, por lo tanto deben estar especificados o de lo contrario se deben aprender mediante algún método [19] [13].

Otra forma de encarar el problema de clasificación puede ser realizar un análisis basado en discriminantes: Primero se especifica una forma paramétrica del límite de decisión (ejemplo, lineal o cuadrática), luego se elige el *mejor* de los límites de decisión especificados a partir de la clasificación de los patrones de entrenamiento. Estos límites de decisión pueden construirse utilizando, por ejemplo, el criterio del *error medio cuadrático*

3.1.3. Reconocimiento Sintáctico o Estructural

En muchos problemas de reconocimiento donde los patrones presentan determinado grado de complejidad, es más apropiado adoptar una perspectiva de jerarquías donde los patrones son vistos como la composición de subpatrones más simples (menor complejidad), siendo estos a su vez construidos por subpatrones aún más sencillos [16][33]. Los subpatrones más sencillos o elementales se denominan *primitivos* y en base a la interrelación de ellos se representan a los patrones más complejos. Existe cierta analogía entre la estructura de patrones y la sintaxis de un lenguaje en el reconocimiento de patrones sintáctico. Los patrones son vistos como frases (u oraciones) que pertenecen a un lenguaje, los primitivos son vistos como el alfabeto de ese lenguaje, y las oraciones son generadas de acuerdo a una gramática. De esta forma, una amplia cantidad de patrones más complejos pueden ser descritos por un numero pequeño de primitivos y reglas gramaticales. A partir de las muestras de entrenamiento se obtiene la gramática para cada clase de patrones.

3.1.4. Redes Neuronales

Las redes neuronales pueden ser vistas como grandes sistemas computacionales en paralelo compuestos por una enorme cantidad de procesadores sencillos interconectados entre si. Los modelos de redes neuronales buscan utilizar algún principio organizacional (como aprendizaje, generalización, adaptabilidad, tolerancia de error y representación distribuida, cálculo) en una red de gráficos ponderados y direccionados donde los nodos son *neuronas* artificiales y los ejes direccionales (con pesos) son las conexiones entre neuronas de entrada y salida. Las características principales de las redes neuronales son la capacidad de aprender complejas relaciones no lineales de entrada-salida, usar procedimientos secuenciales de entrenamiento y la capacidad de adaptarse a los datos.

A pesar de haber descripto brevemente las principales aproximaciones al reconocimiento de patrones en este trabajo se utilizará el abordaje presentado en la seccion 3.1.2, es decir, el reconocimiento estadístico de patrones. Trabajaremos con los patrones representados por un cierto conjunto de características e intentaremos crear fronteras de decisión en el espacio de características que separen a los elementos de distintas clases.

Aproximación	Representación	Función de reconocimiento	Criterio Típico
Template Matching	Muestras, pixeles, curvas	Correlación, medida de distancia	Error de clasificación
Estadístico	Características	Función discriminante	Error de clasificación
Estructural o Sintáctico	Primitivos	Reglas, gramática	Error de aceptación
Redes Neuronales	Muestras, pixeles, curvas	Función de red	Error cuadrático medio

Cuadro 3.2: Modelos de Reconocimiento de Patrones

3.1.5. Resumen

A modo de resumen, se presentan algunos conceptos generales.

- Modelo: representación de un patrón.
- Características o atributos: medidas que componen las representaciones.
- Espacio de representación o de características: el conjunto de todas las representaciones posibles para un cierto problema.
- Aprendizaje: proceso que permite establecer un modelo (establecer los parámetros del modelo o adquirir conocimiento sobre el problema)
- Clase: una clase contiene objetos similares
- Etiquetas: Asignación que se le da a un patrón.
- Reconocimiento de patrones: etiquetado (clasificación, asignación de nombres) de un elemento a una clase determinada.
- Conjunto de entrenamiento: Conjunto de datos utilizado para realizar el aprendizaje.

CAPÍTULO 4

Clases desbalanceadas

En un problema de dos clases se dice que el conjunto de datos se encuentra desbalanceado o sesgado cuando existe una gran diferencia entre la cantidad de representantes de una clase y la otra. El problema del desbalance de clases es muy importante en casos reales donde es costoso clasificar incorrectamente a los representantes de la clase minoritaria, también llamados muestras positivas (a las muestras de la clase mayoritaria se les llama muestras negativas). Por esta razón, en este último tiempo se ha puesto mucho énfasis en desarrollar soluciones para atacar este tipo de problemas. La investigación llevada acabo en esta área se ha basado tradicionalmente en soluciones a nivel de los algoritmos de clasificación y de los datos. Sin embargo en este último tiempo se ha estado trabajando en diferentes estrategias para abordar el problema del desbalance de clases. Estos son:

- Métodos de muestreo para balancear el conjunto de datos.
- Modificaciones en los algoritmos de clasificación utilizados.
- Definición de medidas de performance adecuadas al desbalance de clases.

4.1. Cambios en la distribución de clases

Los métodos de balanceo de clases consisten en “remuestrear” el conjunto de datos original, mediante técnicas de sobre-muestreo de la clase minoritaria y/o de sub-muestreo de la clase mayoritaria hasta obtener un conjunto de datos con un balance entre la representación de las distintas clases. Esta estrategia de “remuestrear” el conjunto de datos presenta algunas desventajas, en especial, el hecho de alterar la distribución original de clases, lo que ha sido muy criticado. Además, en el caso del sub-muestreo se pueden eliminar muestras muy representativas (con mucha información), y en el caso del sobre-muestreo se aumenta de forma artificial el tamaño del conjunto de datos, con el consecuente aumento del costo computacional del algoritmo de clasificación.

4.1.1. Técnicas de Sub-Muestreo

El método más simple e intuitivo es el sub-muestreo aleatorio, el cual es un método que busca balancear la distribución de clases eliminando elementos de la clase mayoritaria de forma aleatoria. A pesar de su simplicidad, se ha demostrado de forma empírica que es uno de los métodos de "remuestreo" más efectivos, aunque, como se dijo anteriormente, con este mecanismo se corre el riesgo de eliminar muestras muy representativas afectando la performance del clasificador. Por otro lado, los métodos heurísticos propuestos se dividen en dos grupos, unos consideran que las muestras que se encuentran cercanas al borde o limite de decisión entre las clases son ruido, mientras que los otros consideran como ruido a las muestras con mayor cantidad de vecinos correspondientes a las otras clases. Un ejemplo del primer grupo es propuesto por Kubat y Matwin en [24] donde introducen una técnica de sub-muestreo que únicamente elimina muestras de la clase minoritaria que son redundantes o que se encuentran en el borde de separación con la clase minoritaria. Batista et al. [2] proponen eliminar tanto muestras de la clase mayoritaria cuyas etiquetas son diferentes a las de sus 3 vecinos más cercanos, como aquellas muestras, también de la clase mayoritaria, que etiquetan incorrectamente muestras de la clase minoritaria. El principal inconveniente de estas técnicas es que no existe un control para eliminar muestras negativas por lo que no garantizan un verdadero balance en la distribución de clases. Por último, cabe destacar que investigaciones en la materia han demostrado que en casos donde el nivel de desbalance es bajo, resulta conveniente aplicar técnicas de sub-muestreo.

4.1.2. Técnicas de Sobre-Muestreo

El sobre-muestreo aleatorio es un método que busca balancear la distribución de clases replicando de forma aleatoria muestras de la clase minoritaria. Por su naturaleza, este mecanismo presenta dos problemas, uno es que al hacer copias exactas de muestras de la clase minoritaria aumenta la probabilidad de que se produzca "overfitting", el otro es que si el conjunto de datos original es grande pero desbalanceado, el tiempo que lleva el proceso de entrenamiento va a aumentar sustancialmente con el sobre-muestreo.

Existen varios métodos de sobre-muestreo basados en **SMOTE** (Synthetic Minority Over-sampling Technique). SMOTE genera nuevas muestras sintéticas de la clase minoritaria interpolando entre muestras positivas cercanas. Al interpolar en lugar de replicar muestras, SMOTE evita el "over-fitting" y hace que el borde de decisión de la clase minoritaria se propague hacia el espacio de las muestras de la clase mayoritaria[4][5]. A pesar de que el sobre-muestreo aumenta el costo computacional, experimentos llevados a cabo por Batista et al. [2] muestran que es conveniente utilizar esta técnica en casos en que el conjunto de datos tiene pocas muestra positivas.

4.1.3. Técnicas Avanzadas de Muestreo

Las técnicas avanzadas de muestreo primero realizan pruebas preliminares de clasificación y luego el re-muestreo. **Boosting** es un algoritmo iterativo que en cada iteración asigna distintos pesos en las distribuciones de entrenamiento. Luego de cada iteración, boosting aumenta el peso asociado a las muestras incorrectamente clasificadas

y disminuye el peso asociado a las muestras correctamente clasificadas de forma independiente. Esto hace que en la siguiente iteración el algoritmo se centre más en las muestras incorrectamente clasificadas[15][27].

4.2. Modificaciones en los algoritmos de clasificación

En la sección anterior se mencionaron algunos de los problemas que presentan las técnicas de remuestreo. Para evitar estos inconvenientes se han propuesto modificaciones en los algoritmos y técnicas tradicionales de clasificación para adaptarlos a los casos en que se tienen conjuntos de datos desbalanceados. Dentro de este grupo vale la pena destacar las técnicas de **manipulación interna del clasificador**, de **aprendizaje sensible al costo**, los **clasificadores de una clase** y la **combinación de clasificadores**.

4.2.1. Manipulación interna del clasificador

Las técnicas de manipulación interna del clasificador buscan predisponer el proceso de discriminación para compensar el desbalance de clases. Barandela et al. [34] propone utilizar una función ponderada de distancia para el clasificador de k-vecinos más cercanos. El objetivo, al utilizar esta función ponderada de distancia, es compensar el desbalance en la etapa de entrenamiento sin modificar la distribución de clases, por lo que para lograrlo, en lugar de asignarle pesos a cada muestra, se le asignan pesos a cada clase. Al ser mayor el peso asignado a la clase mayoritaria, la distancia a las muestras de la clase minoritaria es significativamente menor que la distancia a las muestras de la clase mayoritaria, generando una tendencia sobre las nuevas muestras quienes encuentran a las de la clase minoritaria como su vecino más cercano.

Otra forma de enfrentar el desbalance de clases se logra sesgando el algoritmo SVM de forma tal de alejar lo más posible el hiperplano de separación de clases de las muestras positivas para contrarrestar el efecto generado por el desbalance de clases que "empuja" al hiperplano hacia las muestras positivas. Esto se puede lograr de diversas maneras, por ejemplo en [47] se propone modificar la función de kernel mientras que Veropoulos et. al [24] propone utilizar diferentes constantes de penalización para las distintas clases, dándole un mayor costo al error de clasificación de las muestras positivas que al error de clasificación de las muestras negativas.

4.2.2. Aprendizaje sensible al costo

Incorporar costos en las tomas de decisiones es otra forma de mejorar la performance del clasificador al entrenarlo con una base de datos no balanceada. El modelo de costos se puede expresar en la forma de la matriz de costo, como se puede observar en la tabla 4.1, donde el costo de clasificar una muestra de la clase j como de la clase i corresponde a la entrada λ_{ij} de la matriz. Usualmente esta matriz se expresa en términos del costo promedio de los errores de clasificación. El objetivo al utilizar técnicas de entrenamiento sensibles al costo es minimizar el costo en los errores de clasificación, lo que puede obtenerse al elegir la clase con el mínimo riesgo condicional.

	Class i	Class j
Class i	0	λ_{ij}
Class j	λ_{ji}	0

Cuadro 4.1: Matriz de Costo

Al utilizar una regla de actualización de pesos en cada etapa de entrenamiento se logra una version sensible al costo del método AdaBoost (método de combinación de clasificadores que sera introducido más adelante), mediante esta técnica se le asigna un peso mayor a las muestras positivas incorrectamente clasificadas. MetaCost [22] es otro método de clasificación sensible al costo. Al aplicar un procedimiento sensible al costo, el método aprende de un modelo interno sensible al costo. MetaCost estima la probabilidad de clase utilizando bagging y luego re-etiqueta las muestras de entrenamiento con el mínimo costo de clase esperado para finalmente volver a entrenar utilizando esta base de entrenamiento modificada.

4.2.3. Clasificadores de una clase

Los métodos de clasificación de una clase buscan describir una sola clase de objetos (la clase objetivo), y distinguirla de otros posibles objetos (los "outliers" o clases atípicas). Se asume que se tiene una buena muestra de la clase objetivo, en el sentido que los datos de entrenamiento reflejan el área en el espacio de características cubierto por esta clase. Por su parte, pueden haber pocas o ninguna muestra del conjunto de los outliers. El clasificador de una sola clase debería funcionar utilizando únicamente muestras de la clase objetivo. Un enfoque puede ser utilizar a la clase positiva como la clase objetivo y a la clase negativa como los outliers. Raskutti y Kowlcysk demostraron la ventaja de **One-Class SVM** sobre SVM de dos clases en determinados dominios con desbalance de clases [35]. En particular, demostraron que el entrenamiento con una sola clase es sumamente útil cuando se tienen bases de datos extremadamente desbalanceadas compuestas por un espacio de características sumamente ruidoso.

4.2.4. Combinación de clasificadores

La combinación de clasificadores parte de la base que se tiene un conjunto de clasificadores "débiles" entrenados de forma individual a partir de los cuales se combina la decisión de cada uno al clasificar una muestra nueva. El éxito de la combinación de clasificadores reside en el hecho que se obtiene una precisión de predicción mayor que la del mejor clasificador por si solo. El método **AdaBoost** (Adaptative Boosting) fue introducido por Freund y Schapire [15] en 1995 como un método para crear un clasificador más robusto a partir de un grupo de clasificadores débiles. Este método es adaptativo ya que los clasificadores generados son ajustados hacia las muestras mal clasificadas por los clasificadores previos (de la instancia previa). AdaBoost es sensible a conjuntos de datos ruidosos y outliers, sin embargo en ciertos problemas es menos susceptible al over-fitting que otros clasificadores. Se comienza asignando pesos iguales, pero en cada ronda se aumenta el peso correspondiente a las muestras mal clasificadas forzando a los clasificadores más débiles a concentrarse en estas muestras durante la etapa de entrenamiento. **SMOTEBoost** [5] busca solucionar el problema de over-fitting

que aparece al utilizar la técnica de boosting. En lugar de variar la distribución de los datos de entrenamiento variando los pesos asociados a cada muestra, SMOTEBoost varía la distribución agregando nuevas muestras de la clase minoritaria utilizando el algoritmo SMOTE (introducido cuando se hablo de las técnicas de sobre-muestreo).

4.3. Medidas de performance del clasificador

La gran mayoría de las medidas de performance para problemas de dos clases parten de la matriz de confusión que se observa en la tabla 4.2. A partir de la misma se obtienen 4 simples medidas, TP (True Positive) y TN (True Negative) denotan el número de muestras positivas y negativas correctamente clasificadas, mientras que FP (False Positive) denota el número de muestras negativas clasificadas (erróneamente) como positivas y FN (False Negative) el número de muestras positivas clasificadas como negativas.

Las medidas de performance más utilizadas en los sistemas de aprendizaje son el *error rate* y *accuracy*, definidos de la siguiente manera,

$$err = \frac{FP + FN}{TP + FN + TN + FP}, \qquad acc = \frac{TP + TN}{TP + FN + TN + FP}$$

Sin embargo, se ha demostrado que cuando existe un gran desbalance en la distribución de clases, estas medidas no son apropiadas ya que no toman en consideración los costos de clasificar incorrectamente las distintas clases, son extremadamente sesgadas para favorecer a la clase mayoritaria y son sensibles a los sesgos de clase. Por ejemplo, si tenemos un caso donde el 1 % de las muestras pertenecen a la clase minoritaria, al clasificar todas las muestras como de la clase mayoritaria se obtiene un ”accuracy” del 99 % pero sin lograr clasificar correctamente ninguna muestra de la clase minoritaria.

	Positive	Negative
Positive	TP (True Positive)	FN (False Negative)
Negative	FP (False Positive)	TN (True Negative)

Cuadro 4.2: Matriz de Confusión

Debido a esto, en casos donde se tienen conjuntos de datos desbalanceados, se deben utilizar métricas alternativas para medir de forma independiente la performance al clasificar tanto las muestras positivas como las negativas.

- $TP\ rate = \frac{TP}{TP + FN}$, también denominado Recall o Sensibilidad, es el porcentaje de muestras positivas correctamente clasificadas.
- $TN\ rate = \frac{TN}{TN + FP}$, también denominado Especificidad.
- $FP\ rate = \frac{FP}{FP + TN}$

- $FN\ rate = \dfrac{FN}{FN + TP}$
- $Precision = \dfrac{TP}{TP + FP}$, también llamado pureza.
- $F_{value} = \dfrac{(1 + \beta^2) Recall \times Precision}{\beta^2\, Recall + Precision}$. Combina Precisión y Recall, y permite controlar el peso de cada uno por separado variando el valor de β.
- $MGM = \sqrt{Accuracy_+ Accuracy_-}$, máximo promedio geométrico de la precisión en la clase de la mayoría y de la minoría.

Una métrica muy utilizada también es la curva **ROC** y el área debajo de la curva ROC llamado **(**AUC). La curva ROC es una gráfica de dos dimensiones con el TP rate en el eje de las ordenadas el FP rate en el eje de las abscisas, ésta es utilizada principalmente para comparar costo-beneficio al variar parámetros del clasificador.

CAPÍTULO 5

Selección de características

5.1. Introducción

Mejorar la performance de los clasificadores, eliminar redundancia e información irrelevante son algunas de las razones que justifican implementar métodos de selección de características. Trabajar con representaciones de los patrones en dimensiones elevadas presenta (al menos) dos desventajas, por un lado tienen aparejados costos computacionales desde el punto de vista de los tiempo de ejecución y de la memoria utilizada por los algoritmos, por otro, características irrelevantes llevan a representaciones más complejas de los patrones que pueden afectar el desempeño a la hora de entrenar los clasificadores. La performance de un método de clasificación, dependerá de la relación que exista entre la cantidad de patrones disponibles para el entrenamiento, el número de características y la complejidad del clasificador [20].

A modo de ejemplo, supongamos que se implementa como manera de clasificar patrones el siguiente procedimiento: se parte el espacio de características en celdas y a cada celda se le asigna la etiqueta de una clase (obviamente en función de algún procedimiento que toma en cuenta la distribución de patrones de entrenamiento en cada celda). ¿Qué sucede si aumentamos el número de características? Al incrementar el número de características aumenta naturalmente la dimensión del espacio en que estamos representando a los patrones. El número de celdas (en las cuales dividimos el espacio de características) aumentará de manera exponencial con el número de dimensiones del espacio, si se desea mantener una proporción constante de patrones por celda, la cantidad de patrones debe aumentar en igual proporción que el numero de celdas, por lo tanto la cantidad de patrones debe aumentar de manera exponencial con la características si deseamos mantener la proporción de patrones constante. Este fenómeno es conocido en la literatura como maldición de la dimensión (“curse of dimensionality”) [20].

En conclusión, aumentar el número de características no necesariamente mejora el desempeño de un clasificador y trae aparejados todos los inconvenientes descriptos en

el párrafo anterior. En la literatura se encuentran numerosos artículos que prueban que el aumento de características no mejora en general la precisión de un clasificador, además se observa en la práctica que ocurre el fenómeno contrario: si se aumenta el número de características (utilizando un conjunto pequeño de patrones para el entrenamiento) se observa una caída en la precisión a medida que el número de características aumenta. Este fenómeno es conocido como "peaking phenomenon". Trunk [18] publico en 1979 un ejemplo sencillo que muestra el fenómeno anterior, dicho ejemplo también se puede encontrar resumido en [20].

Establecidas las razones que motivan mantener el número de características acotado, veamos qué métodos disponibles se encuentran para llevar esto a cabo. Supongamos que tenemos en primera instancia p características. El problema se puede describir como: *encontrar* el *mejor* subconjunto de características dentro del conjunto original. Para resolver el problema anterior, debemos enfocarnos en dos aspectos: ¿Cómo *encontrar* el mejor subconjunto? y ¿Qué criterio determina cuál es el mejor subconjunto?

Se abordará la primer pregunta en la sección 5.2 donde se describirán los distintos métodos de búsqueda disponibles y se detallarán aquellos que se decide implementar en este trabajo. Básicamente existen dos enfoques a la hora de establecer métodos de búsqueda: 1) Aquellos métodos que aseguran una solución óptima al problema. Un ejemplo de éstos es el de búsqueda exhaustiva, que consiste en evaluar todos los subconjuntos posibles y seleccionar el *mejor*. El inconveniente que presentan estos algoritmos es que en general son muy costosos desde el punto de vista computacional, a modo de ejemplo, supongamos que tenemos p características y deseamos obtener un subconjunto de d características que mejora la precisión de nuestro clasificador, en ese caso debemos evaluar el clasificador para todas las posibles combinaciones, el número de conjuntos posibles es:

$$n_{conjuntos} = \frac{p!}{(p-d)!d!}$$

Evaluando por ejemplo el número de posibilidades para $p = 28$ y $d = 6$:

$$n_c = \frac{28!}{22!6!} = 376740$$

2)Métodos llamados subóptimos, que no garantizan el mejor subconjunto de características, son mucho más eficientes y menos costosos desde el punto de vista computacional. Dentro de este conjunto de métodos se encuentran el "best first" y "Sequential Forward Selection" qué serán abordados en la sección 5.2.

En referencia a la segunda pregunta que quedó planteada cuando se introdujo el problema, el otro aspecto a tener en cuenta a la hora de implementar una etapa de selección de características concierne a la definición de un criterio que nos permita establecer cual es el *mejor* subconjunto. Se debe explicitar una función de mérito $J(x)$, donde x es un subconjunto de características, que tenga como salida un valor con el cual sea posible evaluar y comparar a los distintos subconjuntos de características y *encontrar* (por alguno de los métodos de búsqueda) aquel que maximiza $J(x)$. Existen distintos criterios para evaluar los posibles subconjuntos de características. Básicamente se pueden distinguir dos tipos de enfoques, por un lado aquellos métodos que utilizan un clasificador para la

evaluación y por el otro lado aquellos que solo toman en cuenta la estructura de los datos de cada clase y cómo se relacionan éstas con los distintos subconjuntos de características. En el segundo enfoque también se tiene en cuenta la correlación o dependencia que existe entre las características. En la sección 5.3 se describirán los distintos métodos de evaluación utilizados.

5.2. Algoritmos de Búsqueda

En esta sección nos concentraremos en los algoritmos que buscan cuál es el subconjunto óptimo de características. La pregunta principal es, a partir de una cantidad d de características, como llegar al "mejor" subconjunto de tamaño p.

La manera natural de realizar esto sería a partir de una búsqueda exhaustiva evaluando la función de mérito $J(x)$ para cada subconjunto posible. El problema de realizar esto es, como se expresó anteriormente, lo costoso que resulta computacionalmente, ya que la cantidad de subconjuntos crece exponencialmente con cada característica. Es por esto que existen otros métodos que intentan reducir el tiempo de búsqueda a expensas de llegar a un subconjunto que puede llegar a no ser óptimo.

Un ejemplo de método de selección que garantiza llegar a un subconjunto óptimo de p características donde p es dado de antemano, de una forma más eficiente que la búsqueda exhaustiva es el "Branch and bound"[45, pag 312]. Este es un método que evita tener que evaluar todos los subconjuntos pero solo se puede aplicar si se cumple que al agregar características $J(X)$ aumenta, es decir, que la función de mérito es monótona con la cantidad de características. Esta es una buena alternativa a la búsqueda exhaustiva, pero es necesario garantizar la condición anterior.

Por otro lado está la evaluación individual de características. Este el método es más simple computacionalmente pero puede llevar a resultados subóptimos . La idea es evaluar la función de mérito J con cada una de las características individualmente, y elegir las *p* mejores. La gran desventaja de este método, es que no evalúa a las características como un conjunto, cuando está demostrado que el desempeño de un subconjunto de características no es la suma de los desempeños individuales. Por ejemplo, 2 características muy correlacionadas pueden dar muy buen desempeño individualmente, pero el incluir a las 2 no aumentaría mucho la discriminalidad, es más, en algunos casos podría ser perjudicial.

Siguiendo con los métodos de búsqueda subóptimos, los siguientes en cuanto a simplicidad son la búsqueda secuencial hacia adelante o hacia atrás. La búsqueda hacia adelante se basa en partir de un conjunto vacío e ir agregando al subconjunto de características una por vez hasta lograr lo que se denomina conjunto final. Supongamos que en un cierto paso tenemos un subconjunto de características al que llamaremos X. El algoritmo evalúa $J(X + \epsilon_i)$ donde ϵ_i es cada una de las características que aún no están en el subconjunto, y elige la que maximiza J. El método se inicia con el conjunto vacío y culmina cuando el agregar cualquiera de las características no mejora la función de mérito más de un cierto umbral que se determina inicialmente. La búsqueda hacia atrás

tiene como única diferencia que se inicia con todas las características y se va sacando de a una. Estos métodos tienen la ventaja que evalúan el mérito de las características como un conjunto, sin embargo al no tener la posibilidad de eliminar un elemento una vez que se agregó (o agregarse una vez que se quitó en el caso de la búsqueda hacia atrás), la elección puede no ser óptima.

Una variante de los métodos anteriores que hace más flexible y profunda la búsqueda, y que por lo tanto nos acercan más al subconjunto óptimo es el *Bestfirst*. Este algoritmo aparte de ir agregando (o quitando) características, guarda la información de los subconjuntos que va obteniendo, y una vez que se llega a la condición de parada (que J no aumente más que cierto umbral) se vuelve a los subconjuntos anteriores, y se examinan otras posibilidades agregando otras características buscando un subconjunto más óptimo.

5.3. Métodos de Evaluación

Como se mencionó en la sección 5.1, para poder elegir un conjunto óptimo de características es necesario definir un criterio que permita medir la habilidad del conjunto para discriminar de forma precisa entre las clases. Esto se logra definiendo una medida de separabilidad de las clases la cual es optimizada con respecto a los posibles subconjuntos. Básicamente existen dos formas bien distintas de encarar este problema:

- Se evalúan las características teniendo en cuenta el clasificador o método de aprendizaje que se va a utilizar para clasificar, este mecanismo se denomina *wrapper*. El proceso llevado a cabo es el siguiente: para cada subconjunto posible de características se diseña el clasificador y se elige el subconjunto para el cual el clasificador logra los mejores resultados de clasificación. En este enfoque el conjunto de características se selecciona según el clasificador a utilizar por lo que se obtendrán diferentes subconjuntos óptimos para diferentes clasificadores. Es muy importante tener esto cuenta, sobre todo en casos donde se utilizan métodos de combinación de clasificadores.

- El otro enfoque consiste en estimar el solapamiento entre las distribuciones en las clases y elegir el subconjunto de características que lo minimiza (es decir, que maximiza la separabilidad). Este mecanismo se denomina *filter*, es independiente del clasificador utilizado y tiene la ventaja de ser relativamente poco costoso de implementar, sin embargo, la desventaja es que al determinar el solapamiento a menudo se realizan supuestos que pueden resultar en una pobre estimación de la discriminabilidad de las clases.

Sería muy sencillo hacer evaluaciones independientes sobre los subconjuntos posibles de características si existiera alguna buena forma de determinar cuándo una característica es relevante para elegir determinada clase. Sin embargo, no existe una medida de relevancia aceptada universalmente por más que se han propuesto varias.

5.3.1. Wrapper

El método de evaluación Wrapper busca el subconjunto de características que maximiza la performance del clasificador, por lo tanto para cada subconjunto de características se debe medir la performance del clasificador. En general el método más utilizado es el de validación cruzada procurando maximizar alguna métrica, por ejemplo, en problemas donde existe un desbalance entre las clases es común buscar maximizar el F_{value} (otros ejemplos de medidas pueden ser vistas en la sección 4.3). Sin embargo existen otros métodos como el estimador de bootstrap que son igualmente validos.

Estudios en esta materia han demostrado que, en términos generales, los métodos de selección recursiva generan subconjuntos más grandes y con mayor precision de clasificación que los métodos de selección *hacia adelante*. Esto se debe principalmente a que la medida de performance es simplemente un estimado, y una simple estimación optimista hará que ambos métodos terminen de forma prematura, los métodos recursivos con subconjuntos formados con demasiadas características y los métodos *hacia adelante* con subconjuntos sin suficientes características. Sin embargo los métodos de selección *hacia adelante* son de mayor utilidad cuando lo que se busca es entender la estructura en la toma de decisiones, ya que generalmente reduce la cantidad de atributos variando poco la performance de clasificación. También se ha visto que generalmente no se justifica el uso de métodos de búsqueda más sofisticados, aunque estos pueden producir mejores resultados en ciertos casos.

Una forma de acelerar la búsqueda es parar de evaluar un subconjunto de atributos desde el momento en que parece poco probable que su performance sea mejor que la de otros posibles subconjuntos. Esto se logra utilizando un test estadístico de importancia en conjunto entre el clasificador basado en el subconjunto en cuestión y los otros clasificadores basados en el resto de los subconjuntos. La diferencia de performance entre dos clasificadores con respecto a determinado subconjunto puede ser -1,0 o 1 dependiendo de si el primer clasificador es peor, igual o mejor que el otro clasificador. Entre las técnicas más utilizadas están *race search* y *schemata search* que pueden ser vistos en [46].

Independientemente de la técnica utilizada, el método de evaluación wrapper no arroja una mejora uniforme de la performance. Debido a la complejidad del proceso, que se ve notoriamente incrementada al incluir el clasificador (feedback constante sobre resultado de clasificación de cada subconjunto) en el método de selección, es sumamente difícil predecir las condiciones bajo las cuales vale la pena utilizar el método.

5.3.2. Filter

Un método simple de selección de características sería utilizar suficientes características para lograr dividir todas las muestras del espacio. Por ejemplo, si solo se utilizan una o dos características, generalmente habrán varias muestras que tendrán la misma combinación de valores o representación para esas características. En el otro extremo, considerar el conjunto total probablemente distinga de forma única las muestras

logrando que no existan dos que tengan los mismos valores (misma representación) para todas las características. Es lógico e intuitivo pensar en seleccionar el menor subconjunto posible capaz de distinguir de forma única las muestras. Esto se puede lograr fácilmente utilizando el método de búsqueda exhaustiva, a un costo computacional muy elevado. Desafortunadamente, esta tendencia hacia una consistencia del conjunto de características en las muestras de entrenamiento es estadísticamente injustificado y puede llevar a overfitting, el algoritmo se puede tornar demasiado complicado para simplemente resolver una inconsistencia causada por el ruido.

En ciertos casos resulta conveniente utilizar técnicas de aprendizaje, independientes del método de clasificación final utilizado, para seleccionar características. Una posibilidad es comenzar utilizando el algoritmo de árboles de decisión sobre el conjunto total de muestras y se seleccionan únicamente las características que realmente se utilizan en el árbol. Aunque esta selección no tendrá efecto alguno si en una segunda etapa (etapa de clasificación) se utilizan para construir otro árbol, si tendrán efecto en un algoritmo de aprendizaje distinto [46]. Por ejemplo, el algoritmo de *vecino más cercano* es muy susceptible a características irrelevantes, y su performance puede ser mejorada usando la técnica de árboles de decisión para la selección de características. El método de vecino más cercano resultante puede incluso tener una mejor performance que el árbol de decisión utilizado para la selección.

Un enfoque distinto es utilizar algoritmos que construyen un modelo lineal, por ejemplo *SVM* lineal, que genera un ranking de las características en función del tamaño de los coeficientes. Una variante un tanto más sofisticada aplica el algoritmo de aprendizaje varias veces. Genera un modelo, lista el ranking de características en función de sus coeficientes, elimina la que se encuentra primera en el ranking y repite el proceso hasta que se eliminan todas las características. Con este método de eliminación recursiva de las características se obtienen mejores resultados para ciertos tipos de bases de datos que simplemente generando un ranking de las características basándose en un modelo único. Es muy importante asegurarse de utilizar la misma escala para medir las características con ambos métodos ya que de no hacerlo, la comparación no tiene sentido (los coeficientes no serían comparables). Esta técnica simplemente genera un ranking de las características por lo que es necesario utilizar otro método para determinar la cantidad apropiada de características a utilizar.

También se puede seleccionar características usando una técnica de aprendizaje basada en las muestras. Se toma un subconjunto aleatorio de las muestras de entrenamiento y se hace una búsqueda de vecinos de la misma clase y de clases diferentes, “near hits” y “near misses”. Si un “near hit” tiene un valor distinto para determinada característica, ésta aparenta ser irrelevante por lo que se disminuye su peso. Si un “near miss” tiene un valor distinto, la característica parece ser relevante por lo que se aumenta su peso. Tras repetir este mecanismo varias veces, se pasa a la etapa de selección donde únicamente serán elegibles aquellas características con peso positivo. Cada vez que se realiza el proceso se van a obtener diferentes resultados debido a la variación en el orden de las muestras. Esto puede evitarse utilizando todas las muestras de entrenamiento y considerando para cada una todos los “near hits” y “near misses”. Este método presenta la desventaja de no detectar características redundantes por estar correlacionadas con

otra/s característica/s. Por otro lado, dos características idénticas serán seleccionados o rechazadas [46].

Otro método para eliminar tanto características redundantes como irrelevantes es seleccionar un subconjunto de características con buena correlación individual con determinada clase pero con poca intercorrelación. La correlación entre dos atributos A y B puede medirse utilizando el *coeficiente simétrico de incertidumbre*:

$$U(A,B) = 2\frac{H(A) + H(B) - H(A,B)}{H(A) + H(B)} \tag{5.1}$$

donde *H* es la entropía funcional. Las entropías están basadas en la probabilidad asociada al valor de cada característica. *H(A,B)*, la entropía conjunta de *A* y *B*, se calcula a partir de la probabilidad conjunta de todas la combinaciones de los valores de *A* y *B*. El *coeficiente simétrico de incertidumbre* se encuentra entre 0 y 1. La evaluación de un conjunto S se hace utilizando

$$\sum j \frac{U(A_j, C)}{\sqrt{\sum i \sum j U(A_i, A_j)}} \tag{5.2}$$

El **numerador** representa la correlación entre las características de S y la clase (capacidad de predecir la clase). El **denominador** representa la intercorrelación media entre características de S (redundancia).

Características irrelevantes son aquellas con pobre capacidad de predicción, mientras que características redundantes son aquellas correlacionadas con una o varias características de S [46].

CAPÍTULO 6

Clasificadores

6.1. One Class SVM

Suport Vector Machine (SVM) fue introducido por Vapnik en la década del 60 y desde esa fecha ha sido ampliamente utilizado y desarrollado. Hoy en día existen variadas aplicaciones que utilizan esta herramienta. El reconocimiento de patrones por medio de SVM ha demostrado ser un método robusto y con excelente desempeño, utilizado para reconocimiento facial [32], reconocimiento de textos [14] y en particular para la detección de fraude en energía eléctrica [30] y [10].

Actualmente, es uno de los métodos predominantes cuando se consultan las publicaciones más recientes relacionadas con la detección de fraude en energía eléctrica, por dicha razón era razonable (e indispensable) estudiar este tipo de técnicas para el caso particular que se pretende abordar. SVM ha demostrado ser una herramienta flexible y versatil. Como contrapartida, estas características terminan siendo una dificultad a la hora de encontrar la implementeación mas adecuada a un problema concreto.

En este trabajo, abordaremos dos tipos de SVM, el primero será One Class SVM[a] y el segundo C-SVM[b]. Ambos métodos consisten en encontrar el hiperplano óptimo, que separa a las clases con el mayor margen posible. La diferencia entre las dos técnicas consiste en que C-SVM considera que existen 2 clases en el problema y en base a las etiquetas de cada consumo busca el hiperplano que mejor separa las dos clases. Por otro lado, One Class SVM como su nombre lo indica, asume que solo tenemos presente una clase (los consumos normales) y lo que se pretende identificar son “outliers”, es decir, consumos que no se ajustan a la clase pero que tomados en su conjunto no presentan necesariamente una estructura de clase. Este segundo método no utiliza ningún tipo de etiquetas (pues estrictamente no hay más que una sola clase).

En las secciones siguientes se estudiarán cada uno de los métodos, se hará

[a] utilizado en [10] para un conjunto muy particular de datos.

[b] este método fue utilizado en [30] con resultados prometedores

un breve desarrollo teórico[c] de las distintas variaciones que presentan y los tipos de parámetros que entran en juego con sus respectivos significados.

Los algoritmos tradicionales de SVM son entrenados usando representantes de cada clase que se pretende separar, por ejemplo asumiendo que estamos en un problema donde se pretende distinguir entre dos clases, se busca un hiperplano que *separe* los representantes de cada clase. Luego se clasifican las nuevas muestras en función del lado que ocupan del hiperplano.

En este método sin embargo, procuraremos extraer información asumiendo que solo tenemos una clase, es decir solo tenemos muestras de una misma clase. Este tipo de técnicas ha resultado importante en varias aplicaciones, a modo de ejemplo, supongamos que queremos identificar un perfil de páginas web de interés para un usuario, en ese caso solo poseemos información de las páginas que visita pero carecemos de información para representar las páginas que no son de su interés.

Scholkopf (1999) fue uno de los pioneros en resolver esta problemática, propuso adaptar la metodología SVM al caso de una sola clase. Luego de transformar las muestras a un espacio de dimensión mayor mediante el uso de un Kernel, se considera el origen como único representante de la segunda clase, para luego aplicar los algoritmos tradicionales de SVM multiclase y buscar el hiper-plano que separa con mayor margen a las muestras del origen.

En otras palabras, se desarrolla un algoritmo que dado un conjunto de muestras, devuelve una función f que toma valor $+1$ en una parte *pequeña* del espacio capturando la *mayoría* de los vectores de muestra y -1 fuera de esa región. La región donde $f(x) = -1$ es considerada la zona donde se encuentras los outliers o *anómalos*.

6.1.1. Algoritmo

Supongamos que tenemos $x_1,\ x_2,\ ...\ x_l\ \in \mathbb{R}^n$ muestras de una clase C, por otro lado consideremos una función $\Phi : \mathbb{R}^n \to \mathcal{H}$ que mapea los datos x_i a un nuevo espacio $\mathcal{H}$ con la propiedad de que $\langle \Phi(x_i),\ \Phi(x_j) \rangle = k(x_i,\ x_j) \quad \forall\ x_i,\ x_j\ \in \mathbb{R}^n$. La propiedad anterior implica que no es necesario transformar cada muestra para realizar el producto interno en el espacio $\mathcal{H}$, es decir podemos expresar dicho producto interno en termino de las muestras en el espacio original (en este caso $\mathbb{R}^n$).

Con el objetivo de separar los datos (en $\mathcal{H}$) del origen, debemos resolver el siguiente problema:

$$\min_{\omega \in \mathcal{H},\, \zeta_i \in \mathbb{R},\, \rho \in \mathbb{R}} \frac{1}{2} \|\omega\|^2 + \frac{1}{\nu\, l} \sum_{i=1}^{i=l} \zeta_i - \rho \tag{6.1}$$

$$\text{sujeto a las restricciones } \langle \omega, \Phi(x_i) \rangle \geq \rho - \zeta_i, \quad \zeta_i \geq 0 \tag{6.2}$$

[c] para un desarrollo teórico más detallado de SVM se recomienda consultar [38] [42] y [7]

Donde $\nu \in (0, 1]$ es un parámetro que regula el compromiso entre cometer errores y obtener mayor separación de la frontera con el origen. Recordemos que el único representante de la segunda clase será el origen de modo que desearíamos que el hiperplano se encuentre lo más lejos posible del mismo. Sin embargo, cuanto más alejamos el hiperplano del origen, más muestras quedaran del lado incorrecto del hiperplano de modo que estaríamos cometiendo errores al clasificar algunas de las muestras x_i.

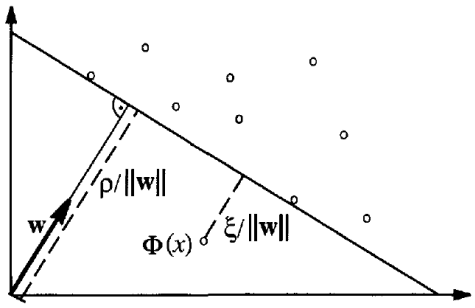


Figura 6.1: Ejemplo en el caso en que trabajamos con muestras en $\mathbb{R}^2$ para ilustrar el compromiso entre obtener un margen amplio y obtener pocos errores. Imagen tomada de [38, pag:232]

Una vez que resolvemos la ecuación (6.1) sujeto a (6.2) obtenemos $f(x) = sign(\langle \Phi(x), \omega \rangle - \rho)$ que nos permitirá clasificar nuevas muestras (como anómalos si $f(x) = -1$ o como normales si $f(x) = 1$).

En este trabajo se utilizarán librerías que tienen implementados los algoritmos para calcular la solución óptima de (6.1) y (6.2), las mismas están disponibles para su descarga en http://www.csie.ntu.edu.tw/~cjlin/libsvm

6.1.2. Elección de los parámetros

Elección del kernel

En la sección (6.1) se presentó en qué consiste el método de One Class SVM como herramienta para detectar muestras anómalas dada una base de datos con estructura de clase única, sin embargo aún queda camino por recorrer antes de tener un clasificador entrenado y listo para su evaluación y posterior uso. En primer lugar debemos determinar el kernel que deseamos utilizar. Tal como se describió en la sección anterior uno de los pasos fundamentales consiste en mapear las muestras originales a un espacio $\mathcal{H}$ dotado de producto interno [39] [30]. La ventaja de esta transformación de las muestras de $\mathbb{R}^n$ a $\mathcal{H}$ mediante Φ consiste en introducir un paso no lineal en el proceso de construcción del clasificador, de este modo podemos obtener funciones de decisión no lineales y trabajar en espacios distintos del espacio original (en el cual la muestras pueden o no ser separables mediante un hiperplano). Teniendo en cuenta lo anterior es evidente la importancia de seleccionar el kernel que iremos a utilizar dado por $k : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R} \quad : \quad k(x, x_i) = \langle \Phi(x),\ \Phi(x_i) \rangle$.

Cualquier función que satisfaga la condición de Mercer [41] puede ser utilizada como función de kernel. En general las funciones de kernel utilizadas en SVM se dividen

en dos categorías: kernels basadas en distancia Euclidea y kernels basadas en el producto interno Euclideana [38]. Los Kernels son seleccionados basados en la estructura de los datos y el tipo de límites entre las clases.

Existen infinidad de kernels propuestos en la literatura (se puede consultar [39] si se desea un análisis profundo en cuanto a la elección de kernels), los más clásicos son:

- Kernels Polinómicos:
$$k(x, x_i) = \langle x, x_i \rangle^d \tag{6.3}$$
- Kernel Gaussiano:
$$k(x, x_i) = e^{-\gamma \|x - x_i\|^2}; \quad \text{con } \gamma > 0 \tag{6.4}$$
- Kernel tanh:
$$k(x, x_i) = tanh(\kappa \langle x, x_i \rangle + \varphi); \quad \text{con } \kappa > 0 \text{ y } \theta \in \mathbb{R} \tag{6.5}$$

En este trabajo se utilizará el kernel Gaussiano tanto para One Class SVM como para C-SVM, pues es uno de los kernels más utilizados en la práctica y además ya fue utilizado en otros trabajos [10][30] de detección de fraude en energía eléctrica con buenos resultados.

Parámetros Óptimos

El paso siguiente luego de determinar el tipo de kernel que se va a utilizar, consiste en determinar los parámetros óptimos que definirán el modelo del clasificador. Estos parámetros dependen naturalmente del kernel que se utilice y del tipo de SVM que se está implementando, en este caso particular se utilizará un kernel Gaussiano y One Class SVM. Por lo tanto los parámetros en juego son ν y γ presentados en (6.1) y (6.4) respectivamente.

Determinar el valor de estos parámetros de una manera adecuada es crucial a la hora de obtener buenos modelos para los clasificadores, en la figura 6.2 se ilustra como varía la frontera de decisión al variar γ. En la figura 6.1 se puede observar como afecta la variación de ν a la hora de determinar el margen y el número de vectores soporte. De qué manera es alterado el desempeño de nuestro clasificador al variar simultáneamente ν y γ no es sencillo de predecir y será abordado en lo que sigue.

6.2. Clasificador C-SVM

6.2.1. C-SVM: Soft Margin

En 1995, Corinna Cortes y Vladimir Vapnik introdujeron la idea de un margen máximo que permita clasificar correctamente ciertas muestras[41]. SVM es un método de clasificación general y su fundamento teórico puede verse en más detalle en [42] y [7]. En caso de no existir un hiperplano tal que pueda separar las muestras en dos clases, el

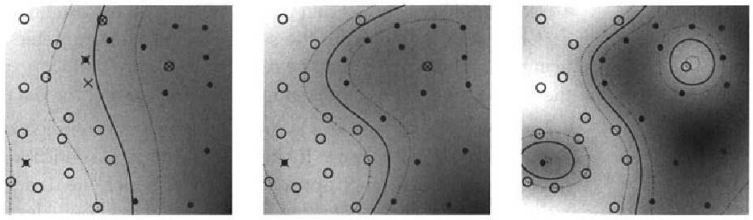


Figura 6.2: Figura a los efectos ilustrativos de como varía la frontera de decisión para un kernel Gaussiano al variar γ. De izquierda a derecha se incrementa γ. Se puede Observar que para valores pequeños de γ la frontera es más lineal y los datos no pueden ser separados sin error, en el otro extremo para valores elevados obtenemos una frontera más ajustada a los datos donde se obtiene un comportamiento no lineal y menos errores en la clasificación. Si reducimos mucho el ancho del kernel debemos tener precauciones de no obtener una curva muy sobre adaptada a los datos de entrenamiento. Imagen tomada de [38]

método de *Soft Margin* elegirá un hiperplano capaz de separar las muestras de la forma más *limpia* posible, y al mismo tiempo maximizando la distancia a las muestras mejor separadas. Este método introduce una variables auxiliares ζ_i que miden el grado de error de clasificación del dato x_i.

El objetivo principal del algoritmo SVM binario (de dos clases) utilizado para clasificación es construir una función de decisión óptima, $f(x)$ que logre predecir de forma precisa a cual de las dos clases posibles pertenecen ciertos datos, minimizando el error de clasificación utilizando

$$f(x) = sgn(g(x)) \tag{6.6}$$

donde $g(x)$ es el límite de decisión entre las dos clases y es determinada a partir de los datos (muestras) de entrenamiento

$$X = x_1, x_2, \ ..., x_n \text{ con } x_i \ \in \mathbb{R}^N, i = 1, 2, ..., n \tag{6.7}$$

donde cada dato de entrenamiento x_i tiene M características y pertenece a una de dos clases

$$Y = y_1, y_2, \ ..., y_n \text{ con } y_i \ \in \{-1, +1\}, i = 1, 2, ..n. \tag{6.8}$$

El límite de decisión entre las dos clases es un hiperplano descrito por la ecuación

$$g(x) = \langle \omega, x \rangle \ + b \, , \tag{6.9}$$

donde ω y b se obtienen de manera tal de clasificar correctamente los datos. Para lograr clasificar correctamente los datos se debe maximizar el margen de separación entre las dos clases. De acuerdo a [7], esto puede ser formulado como un problema de optimización de programación cuadrática

$$\Phi(\omega, \zeta_i) = \text{mín} \left\{ \frac{1}{2} \left\| \omega \right\|^2 + C \sum_{i=1}^{i=n} \zeta_i \right\} \tag{6.10}$$

sujeto a la condición de que todas las muestra de entrenamiento sean correctamente clasificadas (es decir que todas las muestras de entrenamiento se encuentren en el margen o fuera del mismo), esto es

$$y_i(\langle \omega, x \rangle \ + b) \geq 1 - \zeta_i, \ \ i = 1, 2, ..., n \tag{6.11}$$

donde ζ_i para $i = 1, 2, ..., n$ son variables no negativas. Al minimizar el primer término de (6.10) la complejidad del SVM se ve reducida, y al minimizar el segundo término se reduce el número de errores de entrenamiento. El parámetro C en (6.10) es un parámetro de regularización y se preselecciona para ser la compensación entre ambos términos de (6.10).

El problema de programación cuadrática (QP) definido en (6.10) sujeto a (6.11) se resuelve utilizando multiplicadores de Lagrange $\alpha_i \geq 0$ y $\beta_i \geq 0$, y el Lagrangiano

$$L(\omega, b, \zeta, \alpha, \beta) = \frac{1}{2} \left\| \omega \right\|^2 + C \sum_{i=1}^{n} \zeta_i - \sum_{i=1}^{n} \alpha_i y_i [\langle \omega, x \rangle \ + b] - 1 + \zeta_i - \sum_{i=1}^{n} \beta_i \zeta_i \tag{6.12}$$

Según la teoría de optimización de QP, es mejor introducir la formulación del problema dual para resolver (6.12)

$$\max_{\alpha,\beta} W(\alpha, \beta) = \max_{\alpha,\beta} \left\{ \min_{\omega,b,\zeta} (\omega, b, \zeta, \alpha, \beta) \right\}. \tag{6.13}$$

La solución óptima a este problema se obtiene, primero minimizando con respecto a ω, b y ζ, y luego maximizando con respecto a $\alpha \geq 0$ y $\beta \geq 0$. Al introducir (6.12) en (6.13), se lleva el problema al dual,

$$\max_{\alpha} \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j \langle x_i, x_j \rangle \tag{6.14}$$

Esta formulación del problema dual es maximizada sujeta a la restricción

$$\sum_{i=1}^{n} \alpha_i y_i = 0 \ \text{ y } \ 0 \leq \alpha_i \leq C \text{ para } i = 1, 2, ..., n \tag{6.15}$$

El vector solución ω se expresa en función de las muestras de entrenamiento, pero solamente de aquellas para las que se cumple que $\alpha_i \neq 0$. Estas muestras de entrenamiento cumplen la condición de Karush-Kuhn-Tucker (KKT)

$$\alpha_i y_i [\langle \omega, x \rangle \ + b] - 1 + \zeta_i = 0, \ \ i = 1, 2, ..., n \tag{6.16}$$

La ecuación(6.16) manifiesta que para describir el hiperplano de separación, únicamente se necesitan los vectores de entrenamiento correspondientes a multiplicadores de Lagrange distintos de cero, es decir los **vectores soporte**. En consecuencia, se obtiene que la expresión del límite de decisión $g(x)$ está determinado únicamente por un subconjunto de los datos de entrenamiento

$$g(x) = \sum_{i=1}^{N_s} \alpha_i \ y_i \langle x, x_i \rangle + b \tag{6.17}$$

donde x es el vector de entrada, $\langle x, x_i \rangle$ es el producto interno, N_s es el número de vectores soportes y b es el sesgo.

Como se mencionó en la sección 6.1.2 en la mayoría de los casos no es posible determinar un límite de decisión lineal. En estos casos SVM mapea el vector de entrada x a un espacio de características de mayor dimensión $\mathcal{H}$ dotado de producto interno [42] [7]. Esto se logra introduciendo una función de Kernel $K(.,.)$,

$$\langle x_i, x_j \rangle \rightarrow K(x_i, x_j) \tag{6.18}$$

Al introducir esta modificación en (6.14) se obtiene lo siguiente

$$\max_{\alpha} \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j K(x_i, x_j) \tag{6.19}$$

y se maximiza sujeta a las condiciones en (6.11)

Al introducir esta función de Kernel, el límite de decisión en (6.17) se transforma en

$$g(x) = \sum_{i=1}^{N_s} \alpha_i \, y_i K(x, x_i) \; + b \; . \tag{6.20}$$

6.2.2. CS-SVM: SVM Sensible al Costo

En un problema de dos clases se dice que el conjunto de datos se encuentra desbalanceado o sesgado cuando existe una gran diferencia entre la cantidad de representantes de una clase y la otra. El problema del desbalance de clases es muy importante en casos reales donde es costoso clasificar incorrectamente a los representantes de la clase minoritaria. Un caso de estos es la detección de fraudes

En la sección 4.2.2 se presentó la técnica de aprendizaje sensible al costo. La idea detrás de este método es incorporar costos en las tomas de decisiones procurando mejorar la performance del clasificador al entrenarlo con una base de datos no balanceada.

Si recordamos como formulamos el problema de C-SVM (6.10) vemos que se busca minimizar

$$\Phi(\omega, \zeta_i) = \text{mín} \left\{ \frac{1}{2} \left\| \omega \right\|^2 + \sum_{i=1}^{i=n} C_i \zeta_i \right\} \tag{6.21}$$

sujeto a la condición (6.11).

En SVM, C_i es el mismo para todos los patrones, sin embargo en CS-SVM este factor es utilizado para representar un peso distinto a cada clase y así incorporar un costo en la toma de decisiones. Por lo tanto la ecuación (6.21) queda

$$\Phi(\omega, \zeta_i) = \text{mín} \left\{ \frac{1}{2} \left\| \omega \right\|^2 + \sum_{i/y_i=1} C^+ \zeta_i + \sum_{i/y_i=-1} C^- \zeta_i \right\} \tag{6.22}$$

sujeto a (6.11).

La resolución de este problema es idéntica a la vista para C-SVM, salvo que ahora los multiplicadores de Lagrange tendrán cotas superiores diferentes,

$$C^+ \geq \alpha_i \geq 0, y_i = +1 \tag{6.23}$$

$$C^- \geq \alpha_i \geq 0, y_i = -1 \tag{6.24}$$

y el límite de decisión sigue siendo (6.17).

Si durante el entrenamiento utilizamos $C^+ > C^-$ el plano de clasificación se va a volcar hacia la clase negativa (las muestras normales) haciendo que más patrones caigan del lado de la clase positiva, dificultando una clasificación incorrecta de la clase positiva.

6.2.3. Elección de los Parámetros

Método de selección de los Parámetros Óptimos

La selección de los dos parámetros del modelo C-SVM es determinante para la precisión de clasificación. Por lo tanto, el modelo óptimo del clasificador se obtiene optimizando el parámetro del kernel RBF, γ, y el parámetro de penalización de error C. El primer paso para determinar los parámetros consiste en dividir la base de datos en dos, una de entrenamiento y otra de test. La base de entrenamiento es utilizada para probar distintos valores de C y γ y así obtener el modelo óptimo del clasificador.

En la practica, ambos parámetros se eligen utilizando validación cruzada. Con este propósito, se divide la base de entrenamiento en p partes iguales y se realizan p corridas de entrenamiento. En cada corrida se utilizan $p-1$ conjuntos para entrenar el clasificador (generar un modelo) y se utiliza el restante como un conjunto independiente de validación para optimizar los parámetros. En el caso más simple, se eligen los parámetros que en promedio generan el mejor modelo según determinado criterio en las p corridas. Una vez determinados estos mejores parámetros, se entrena el clasificador con toda la base de entrenamiento. Esta metodología presenta ciertos problemas. Primero, se utiliza la misma base para optimizar los parámetros y para entrenar el clasificador, esto puede llevar a overfitting. Segundo, la configuración óptima de parámetros para conjuntos de datos de tamaño m y $9/10m$ generalmente no coinciden. Por lo general, el conjunto de datos de menor tamaño requiere una mayor regularización. Esto puede significar un Kernel Gaussiano más amplio, un menor grado polinomial, un menor C o un ν mayor. Aun peor, es teóricamente posible que exista una fase de transición en la curva de aprendizaje entre conjuntos de diferente tamaño. Esto significa que la generalización de error como una función del tamaño del conjunto puede variar drásticamente entre $9/10m$ y m. A pesar de todo lo antes dicho, en general no se toman en cuenta esta consideraciones teóricas y se utilizan estos parámetros con excelentes resultados [38].

Al utilizar CS-SVM, se introduce una variable más en la búsqueda de los parámetros óptimos y el problema consiste en encontrar C^+, C^- y γ óptimos. Aumentar la cantidad de variables en la búsqueda del modelo óptimo naturalmente aumentará el

costo computacional del problema. Un método para disminuir este efecto es proponer una cierta relación entre C^+ y C^-, volviendo el problema a uno de dos variables. En [30] por ejemplo, se propone utilizar pesos de clase definidos como la relación entre el número de muestras de entrenamiento sobre el número de muestras de cada clase.

6.3. Optimun Path Forest (OPF)

Uno de los papers reseñados en la bibliografía acerca de la técnica de reconocimiento de patrones aplicada a la detección de fraudes en el consumo de energía eléctrica fue "A New Approach for Nontechnical Losses Detection Based on Optimum-Path Forest" de Caio César Oba Ramos et al. [9]. En el mismo se detalla un clasificador llamado Optimum-Path Forest (OPF en adelante) con el cual se obtienen muy buenos resultados en un problema de similares características al que se pretende abordar.

La técnica de OPF surge como una alternativa, al igual que SVM, para los problemas donde las clases no son linealmente separables. Como ya se vió, SVM asume una separabilidad en una dimensión mayor a la que tienen los datos, pero las operaciones que lleva a cabo el mismo para lograr esto son muy costosas en cuanto a tiempo, sobre todo cuando el conjunto de entrenamiento es grande. OPF intenta solucionar el problema de clases solapadas mediante un método computacionalmente más eficiente.

6.3.1. Descripción del método

A continuación se realizará una breve descripción del método, un análisis más a fondo del mismo se puede encontrar en [40].

Lo primero a realizar, al igual que en los métodos anteriores, es dividir los datos en un conjunto de entrenamiento y uno de test, a quienes llamaremos Z_1 y Z_3 respectivamente. Con los vectores de Z_1 se crea el clasificador y con Z_3 se mide su performance, manteniéndose la información de a qué clase pertenecen los elementos de test, ocultas hasta el final. La técnica de OPF se encuentra dentro de los clasificadores llamados supervisados, es decir, para entrenarse, usa las etiquetas de los consumos.

En la fase de entrenamiento lo que se intenta es crear un grafo donde los nodos son los vectores pertenecientes a Z_1. En un principio se conectan todos los nodos entre sí mediante arcos $A = Z_1 \times Z_1$ lográndose una red en donde cada elemento de Z_1 está conectado a todos los elementos del conjunto (ver figura 6.3-a). Llamaremos camino a una secuencia de elementos distintos $\pi = \langle s_1, s_2, ..., s_k \rangle$, donde $(s_i, s_{i+1}) \in A$ para $1 \leq i \leq k-1$. A cada uno de estos caminos π se le asigna un costo $f(\pi)$ dado por la función de costo f. Un camino se dice óptimo si $f(\pi) \leq f(\pi')$ para cualquier otro camino π' entre los mismo elementos. Papa et al. en [40] recomiendan usar la función de costo f_{max} la cual a cada camino le asocia como costo la distancia máxima entre elementos del mismo que compartan un arco (elementos adyacentes).

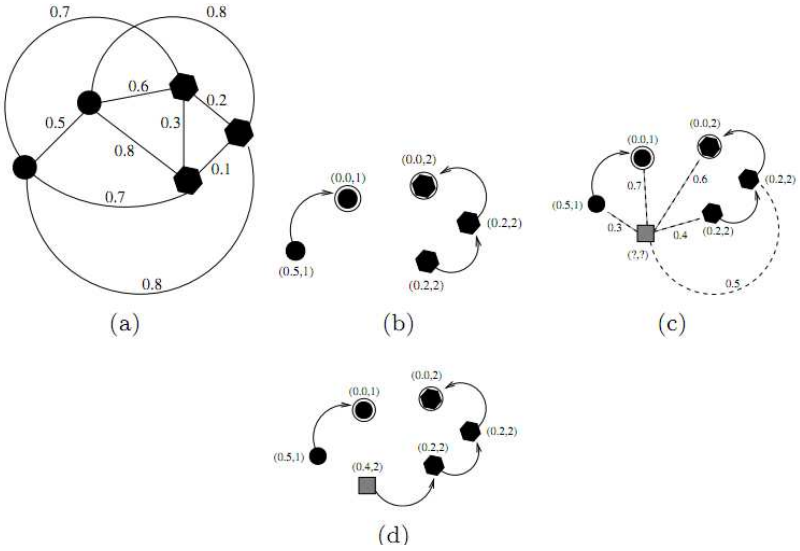


Figura 6.3: (a) El grafo inicial completo con todos los elementos de Z_1 y las distancias. (b) El optimum-path forest para el caso de (a) y 2 prototipos (nodos marcados con círculos). (c) Patrón de test (cuadrado gris) y las conexiones a los patrones de entrenamiento. (d) Camino óptimo al prototipo

Entrenamiento

Antes de detallar cómo es el entrenamiento del clasificador vamos a introducir el concepto de "prototipo". Los prototipos son elementos pertenecientes a Z_1 que van a representar a cada clase. La idea es que el resto de los elementos de la clase se asocien a uno de estos prototipos. La importancia de estos elementos puede trazar una analogía con los vectores soporte en SVM.

El algoritmo de entrenamiento parte del grafo completo, (fig 6.3-a) y lo que intentará es minimizar $f(\pi)$ para todo camino desde los prototipos a cualquier elemento de Z_1, dejando solo los arcos que componen algún camino óptimo. Diremos que el grafo es óptimo si la suma de los pesos de los arcos (distancia entre elementos que une) es mínima comparada con cualquier otro grafo resultante posible. Si resulta así le daremos el nombre de optimum-path forest. Por más información acerca de cómo se logra el grafo óptimo puede consultarse [40].

El conjunto óptimo de prototipos está formado por los elementos de Z_1 adyacentes que poseen clases distintas(figura 6.3-b) [40]. Antes de comenzar el entrenamiento, se eligen estos prototipos y se crea el Optimun Path Forest.

Clasificación

Una vez que el algoritmo está entrenado podemos ingresarle una nueva muestra para que la clasifique en una de las clases de nuestro problema. Sea t una muestra cualquiera de Z_3. Lo primero a hacer es considerar todos los arcos que conectan a t con muestras de Z_1 como si t fuese parte de grafo inicial (Figura 6.3-c). Se consideran

todos los caminos posibles desde los prototipos a t, y se busca cuál de todos estos es el óptimo. Para esto se calcula el costo de cada uno de estos caminos y se elige el que lo minimiza. Por último se etiqueta a t con la clase del prototipo asociado al camino óptimo (figura 6.3-d). De esta manera se clasifican todos los elementos de Z_3. La performance del clasificador se desprenderá de comparar las etiquetas previas con la salida del clasificador.

Aprendizaje

Los autores de este método también proponen una mejora a lo dicho anteriormente, se trata de que el clasificador "aprenda" cuáles son los elementos que aportan más información de cada clase y entrenar con ellos. Lo que se hace es, primero dividir la base de datos en 3 conjuntos Z_1, Z_2 y Z_3, de entrenamiento, aprendizaje y test respectivamente. El clasificador se entrena con los elementos de Z_1 de la misma manera que la descrita previamente, logrando así el Optimum Path Forest. La diferencia es que antes de pasar al test, se clasifican los elementos de Z_2 cual si fuera el conjunto de test, y los elementos mal clasificados de este conjunto se intercambian aleatoriamente por elementos de Z_1. Esto se basa en la suposición de que los elementos mal clasificados son los elementos con más información de las clases. Se vuelve a entrenar con el nuevo conjunto Z_1, y se realiza esta iteración una determinada cantidad de veces. Al finalizar se elige el conjunto Z_1 definitivo, que será el que originó menor error en los elementos de Z_2.

Es importante destacar en este método que el entrenamiento y la clasificación se hace de forma automática, libre de cualquier parámetro. La ventaja de esto, es que a diferencia de SVM no hay que hacer validaciones cruzadas dentro del conjunto de entrenamiento ni probar variando los parámetros para descubrir cuales son los óptimos. Por otro lado tiene una desventaja, es más difícil enfocar el clasificador a un problema de clases desbalanceadas y con distintos costos de error, por ejemplo, si nuestro objetivo es minimizar los anómalos mal clasificados.

Un tema que no es menor y que hay que tener en cuenta es cómo se define la función distancia entre elementos. En este método la noción de distancia es fundamental, ya que la clasificación se hace a partir de ella. Cuando se está mirando el espacio de características, puede que cada característica en particular tenga una distribución en el espacio distinta y por lo tanto la distancia euclideana no represente de la mejor manera la separación entre patrones.

6.4. Arboles de decisión

Una forma intuitiva de clasificar un elementos sería a partir de una secuencia de preguntas en la cual se parte de una pregunta inicial y, dependiendo de la respuesta de ésta, se continua el procedimiento hasta que se llega a una conclusión sobre qué etiqueta asignarle. En esta idea se basan los árboles de decisión, se parte de un nodo raíz con todos los elementos y se realiza una primera división a partir del valor de una característica. De este nodo nace por consiguiente una rama para cada respuesta posible.

Los elementos se dividen según la rama que les corresponda y se forman subnodos. Se repite el procedimiento hasta que llega a un nodo terminal, al cual se le asigna una etiqueta. A la hora de clasificar un nuevo elemento, se comienza por el nodo raiz, se le realizan la primera pregunta y se le asigna el subnodo que corresponda. Se repite el procedimiento hasta que llega a un nodo terminal, y en ese momento se le asigna la etiqueta de ese nodo. La noción general del árbol se puede observar en el ejemplo que nos acerca Duda et al. en [12]. En el mismo se pretende clasificar distintas frutas a partir de determinadas características (nominales en este caso) partiendo del nodo raíz y comenzando por la característica “Color”

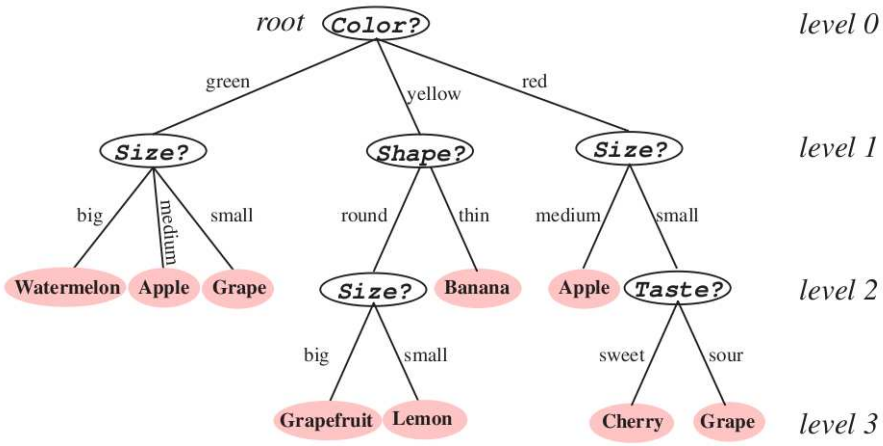


Figura 6.4: Ejemplo de árbol con características no nominales

Como sostiene Kuncheva et al. en [23] existen 3 items principales que describen a los árboles y que a su vez los distinguen de los demás algoritmos de clasificación,

- Si todos los elementos son distinguibles, es decir no existen elementos idénticos pertenecientes a clases diferentes, siempre se puede construir un árbol con error de clasificación cero. Esto tiene como consecuencia que sea un clasificador muy inestable, ya que al memorizar tan bien los elementos de entrenamiento, y así no cometer ningún error con ellos, pequeños cambios en estos elementos conducen a importantes cambios en la estructura del clasificador

- Como se mencionó al comienzo, los árboles son una forma intuitiva de clasificar, ya que el proceso de decisión es facilmente seguible al ser una serie de preguntas concretas.

- Tanto las características cualitativas como las cuantitativas son viables en los árboles de decisión. Los árboles son clasificadores no métricos, es decir no usan ningún tipo de métrica o distancia para clasificar. Esto es realmente importante para que se puedan usar ambos tipos de características.

Hasta el momento se han visto algunas generalidades de los árboles pero aún nos quedan algunas preguntas claves por contestar. En el proceso de creación del árbol se parte del nodo raíz y se elige alguna característica para hacer la primera división en los subnodos y así sucesivamente hasta que se le coloca una etiqueta a cada nodo terminal cuando todos los elementos de estos nodos pertenecen a una sola clase o se consideran

suficientemente “puros”. La primera cuestión es cómo elegir la característica que divide cada nodo, y la segunda es cuándo consideramos que un nodo es suficientemente “puro”. A esto le llamaremos criterio de crecimiento y criterio de parada respectivamente.

Como se dijo anteriormente, los árboles son muy sensibles al conjunto de entrenamiento, por lo que siempre hay un riesgo muy grande de “overfiting”, es decir de sobreadaptación del clasificador a los datos de entrenamiento. Esto puede hacer que la performance del árbol en el conjunto test se vea sensiblemente afectada. Para esto es que una vez que creamos el árbol es muy común que se le aplique una técnica de podado, es decir, se eliminen los nodos de niveles más bajos, y a costas de aumentar el error durante el entrenamiento, se mejore el desempeño en el conjunto de test.

6.4.1. Crecimiento del árbol

Lo primero a definir antes de construir el árbol es si se van a utilizar divisiones binarias, o no binarias, es decir si de cada nodo deberán salir necesariamente 2 ramas o pueden salir más. En general se tiende a elegir árboles binarios, ya que cualquier nodo con decisiones múltiples se puede escribir como combinaciones de decisiones binarias. Para el caso de características continuas, la pregunta para la decisión binaria sería del estilo “¿Es $x < x_s$?”, donde x_s es el umbral para la decisión en ese nodo.

Cuando comenzamos a construir el árbol, desde el nodo raíz, y luego en cada subnodo que se forme, nos vamos a enfrentar con la pregunta de qué característica seleccionar para realizar la división y en el caso de que sea una característica continúa, cuál es el umbral con el cual se decidirán los nodos hijos. Lo que se intentará buscar son los parámetros que mejor separen a las clases, o dicho de otra manera, los que generen subnodos más puros. Para esto es necesario definir lo que llamaremos pureza en un nodo, o en nuestro caso lo que llamaremos impureza.

La idea del concepto de impureza es crear una función que sea mínima (cero) cuando todos los elementos de un nodo son de la misma clase (condición deseable) y que sea máxima cuando hay la misma cantidad de elementos de cada clase, lo cual representa el escenario menos deseable. Consideremos un problema de c clases $\Omega = \{\omega_1, ..., \omega_c\}$ y sea P_j la proporción de elementos de la clase j en un cierto nodo de un árbol de clasificación. Definiremos algunas de las medidas de impureza más comunes.

Impureza de información (entropía)

Esta medida de la impureza está basada en el concepto de entropía, o sea de cuanta información contiene ese nodo. La formula en cuestión es la siguiente:

$$i(N) = -\sum_{j=1}^{c} P_j log P_j$$

Se puede comprobar facilmente como cuando existen elementos de una sola clase $i(N) = 0$ (todos los P son 0 salvo el de la clase de los elementos que vale 1) y es máxima

cuando todas las clases tienen la misma cantidad de elementos, en que $i(N) = log(c)$. Duda et al. en [12] afirma que la impureza de información es la medida más popular.

Impureza de Gini

La impureza de Gini se calcula de la siguiente manera,

$$i(t) = 1 - \sum_{j=1}^{c} P_j^2$$

Al igual que la entropía de información vale 0 cuando todos los elementos son de una misma clase y tiene su máximo cuando las etiquetas son equidistribuidas entre las clases.

Impureza de error de clasificación

Esta medida de impureza representa el error de clasificación que se cometería si se le asigna al nodo la clase de la mayoría. Se calcula de la siguiente forma,

$$i(t) = 1 - max_{j=1}^{c}\{P_j\}$$

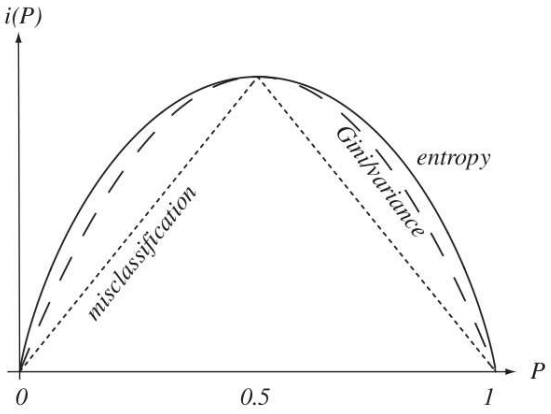


Figura 6.5: En esta imagen se ilustra para un problema de dos clases distintas medidas de impureza, en función de la proporción de elementos de cada clase. Imagen tomada de [12].

Supongamos ahora que ya se eligió una manera de medir la impureza de un nodo y nos enfrentamos al problema de cómo hacer la división, es decir, cuál va a ser la pregunta que va a dividir los datos en las 2 ramas. Al nodo donde nos encontramos lo llamaremos t y a los nodos hijos t_R y t_L. Lo que intentaremos es maximizar la disminución de la entropía entre t y los nodos hijos. A este cambio de entropía lo representaremos como $\Delta i(t)$ y lo calcularemos como,

$$\Delta(i) = i(t) - P(X = 0)i(t_L) - P(X = 1)i(t_R)$$

En este caso X es la característica (binaria) usada para la división. Si la característica fuese continua, deberíamos calcular también el umbral óptimo. $P(\zeta)$ es la probabilidad de que ocurra el evento ζ.

6.4.2. Criterios de parada

El crecimiento del árbol puede extenderse hasta que no existan nodos terminales con impurezas. Como se expresó antes, en el caso de que no existan elementos idénticos asociados a distintas clases, se puede construir un árbol perfecto con impureza cero. De todas maneras, como también se expresó antes, este sería un árbol sobre-entrenado, ya que estaría muy adaptado a los datos de entrenamiento y es muy probable que no tenga una buena performance con nuevos datos. Para evitar esto una de las soluciones es detener el crecimiento antes de que se llegue a los nodos puros. La cuestión es cuándo detenerse, si nos detenemos muy tarde estaremos sobre-entrenando, si frenamos muy temprano estaríamos perdiendo poder de clasificación. Duda et al. en [12] propone los siguientes métodos:

- Extraer del conjunto de entrenamiento un set de datos de validación para testear el árbol a medida que lo vamos construyendo. Una vez que la performance deje de aumentar se detiene el crecimiento
- Establecer un umbral para la reducción de impureza $\Delta(i)$, es decir, cuando se plantee en un nodo la división en 2 nodos hijos, sólo hacerlo si la reducción de impureza está por encima de cierto umbral β. Todavía queda la pregunta de cuál es el mejor parámetro β, o sea, este método no nos asegura parar en el momento óptimo
- Establecer un número mínimo de elementos en un nodo. Si llegamos a un nodo con una cierta cantidad de elementos k, se decide no seguir dividiendo, ya que se supone que a partir de ese punto se estará sobre-entrenando.
- Penalizar la complejidad (tamaño) del árbol agregando este factor a la reducción de impureza, de tal manera, que cuando una división aumentó la complejidad y no reduzca suficientemente la impureza, se decida no dividir. Un ejemplo puede ser,

$$\text{minimizar} \quad \alpha \times n^{\circ}\_de\_nodos + \sum_{nodos} i(t)$$

- Usar test de hipótesis para saber si una división es útil o no. La idea de este criterio es comparar la distribución de las clases en un nodo con el de los nodos hijos. Si con una cierta confianza la distribución de los nodos hijos es la misma que la del nodo padre, significa que no vale la pena dividir y debemos parar en ese punto.

6.4.3. Métodos de podado

A veces puede suceder que cuando utilizamos algún método de parada, éste detiene el crecimiento antes del punto óptimo, es decir, se pierden ramificaciones que

serían muy positivas para el algoritmo. A esto se le llama efecto horizonte. Por esta razón se utiliza la opción de podado, es decir, se deja crecer el árbol y luego se eliminan las ramificaciones no beneficiosas. El podado, como alternativa a los criterios de parada, tiene la ventaja que tiene más visión del problema, pero es computacionalmente más costoso, incluso si el conjunto de entrenamiento es muy grande, puede que sea prohibitivo. Algunos de los métodos que existen para el podado son los siguientes.

REP (Reduced Error Pruning)

Este el método más simple de podado, en el cual se usa una porción de los datos (base de podado) para realizar el mismo. Una vez que se crea el árbol, proceso en el cual la base de podado permanece oculta, se evalúa el error con ella. Posteriormente, comenzando por los nodos terminales y siguiendo hacia arriba, se van eliminando las últimas ramificaciones y calculando el error de clasificación. En el caso que el error disminuya al convertir un nodo intermedio en un nodo terminal, se realiza este cambio y se eliminan las ramificaciones inferiores. Es bueno hacer notar que para poder clasificar, lo que se hace es asignarle a los nodos intermedios, cuando se convierten en terminales, la etiqueta de la clase mayoritaria.

Podado de Error Pesimista

A diferencia de lo que sucede en REP, en este método no se utiliza un conjunto independiente de datos sino que se utiliza el conjunto de entrenamiento para realizar el podado. El árbol mas popular hoy en día, C4.5, realiza un podado de error pesimista luego de crear el árbol. Se realizará una breve explicación del método, un analisis más a fondo se puede encontrar en [28]. En este método se recorre el árbol desde el nodo raíz hacia los subnodos y se va evaluando si hay que podar o no. Para esto lo primero que se hace es estimar el error de entrenamiento en cada nodo como $r(t) = \frac{e(t)}{N(t)}$ siendo $e(t)$ la cantidad de patrones mal clasificados en ese nodo [d] y $N(t)$ la cantidad total de elementos en ese nodo. $r'(t)$ es la corrección de continuidad y la definimos como $r'(T_t) = \frac{e(t)+1/2}{N(t)}$. Generalizamos esta idea para los subárboles (árboles que tienen como nodo raíz a cada nodo y contienen todas sus ramificaciones) como $r'(T_t) = \frac{\sum(e(t)+1/2)}{\sum N(t)}$ donde la suma es en todos los nodos del subárbol y T_t se refiere al subárbol del nodo t. En el conjunto de entrenamiento siempre se cometen menos errores que en un conjunto de prueba, por lo que esta estimación del error es optimista. Lo que vamos a hacer es suponer que los patrones son generados por un proceso de Bernoulli y por lo tanto vamos a decidir a podar el subárbol si,

$$r(t) \leq r'(t) + \sqrt{(\frac{r'(T_t)[n - r'(T_t)]}{n})} - \frac{1}{2}$$

Esta fórmula se justifica en tomar una opción pesimista para el intervalo de confianza del cálculo del error, sin embargo [28] y [23] aseguran que la deducción de esta fórmula es estadísticamente dudosa.

[d] Esto suponiendo que se le asigna a cada nodo la etiqueta de la clase mayooritaria

CAPÍTULO 7

Combinación

La filosofía detrás de los métodos de combinación de clasificadores es obtener un esquema final cuya performance sea superior a la de los clasificadores que estamos combinando. Si bien a simple vista la afirmación anterior parece obvia, en muchos casos puede suceder que un método de combinación de clasificadores empeore el desempeño que se obtenía al utilizar individualmente uno de los clasificadores disponibles . Por otro lado, aún cuando se mejora el desempeño y disminuye el porcentaje de error al combinar clasificadores, no se puede perder de vista el costo computacional y el aumento en la complejidad del algoritmo final. La mejora en el desempeño debe ser tal que se justifica la carga y complejidad que se esta agregando al sistema. Otra precaución que se debe tener en cuenta antes de incursionar en cualquier técnica de combinación de clasificadores, consiste en tener cabalmente analizados y probados los cimientos sobre los que se basará la construcción del combinador. En otras palabras, de nada serviría estudiar a fondo el modo de combinar clasificadores si estos no se adaptan al problema, o las características propuestas no son las adecuadas, o no se implementaron los métodos de selección y/o extracción de características convenientes.

Una vez que se tiene cierta convicción que las características y el conjunto de clasificadores se adaptan al problema que se pretende abordar, y que además, ya fueron ajustados los parámetros de cada clasificador de manera adecuada, el siguiente paso es encontrar la manera de combinar la información que aporta cada algoritmo para obtener un esquema final más robusto y con mejores desempeños. Dietterich [11] sugiere tres tipos de razones por las cuales un conjunto de clasificadores combinados es mejor que un clasificador individual:

1. Estadísticas
   Supongamos que se tienen L clasificadores del tipo vecino más cercano, construidos en base a submuestreos de una base de datos. En lugar de elegir uno de los clasificadores, se puede considerar usarlos todos promediando las salidas o implementando un esquema de voto por mayoría. El nuevo clasificador (obtenido a partir de la combinación de los L clasificadores indivduales) puede que no mejore

la performance del mejor de los clasificadores originales, sin embargo disminuirá el riesgo de elegir el clasificador incorrecto dentro del conjunto de L clasificadores.

2. Computacionales
   Algunos clasificadores necesitan, mediante entrenamiento, ajustar uno o más parámetros antes de estar listos para clasificar. Para esto se suelen utilizar algoritmos de maximización que, dependiendo del punto de partida, pueden llegar a soluciones que corresponden a máximos locales. Mediante la combinación de clasificadores entrenados desde distintos puntos de partida, se puede obtener un clasificador más cercano al óptimo.

3. Generalización
   Mediante la combinación de clasificadores podemos obtener nuevos clasificadores, los segundos se encuentran en un espacio más amplio que contiene a los primeros. A modo de ejemplo, supongamos que para un problema dado el clasificador óptimo es no lineal, pero sólo estamos considerando clasificadores base lineales. El clasificador óptimo se encuentra fuera del espacio de los clasificadores que estamos considerando, sin embargo mediante la combinación de clasificadores lineales podemos aproximar a un clasificador no lineal que se ajuste mejor al problema.

Además de las ventajas que puntualiza Dietterich en [11], la combinación de clasificadores tiende a generalizar el clasificador final desembocando en una solución más robusta y menos adaptada a las bases que se están utilizando para el entrenamiento. Para lograr lo anterior, en general se requiere que los clasificadores base que se están combinando posean cierta independencia que se puede alcanzar a distintos niveles. A nivel de la base de datos, una estrategia posible puede ser *dividir*[a] la base de entrenamiento en varias bases de datos y utilizar cada una de estas para cada uno de los clasificadores base que se desea emplear. Otra estrategia puede ser plantear clasificadores diferentes con naturalezas distintas y que pretendan atacar o concentrarse de manera distinta en los datos disponibles, a modo de ejemplo, combinar un clasificador de tipo SVM con uno de vecino más cercano, prestando más atención al segundo cerca de la frontera del SVM y viceversa. Otra manera de obtener cierta independencia en los clasificadores base, que redunda en una generalización mayor y menos sobre ajuste, es utilizar distintas características para los distintos clasificadores base, de este modo aún cuando las bases de datos de entrenamiento coincidan, los clasificadores base serán menos dependientes.

7.1. Fusión y Selección de Clasificadores

Entre todos los métodos de combinación de clasificadores que se pueden encontrar en la literatura, generalmente se pueden agrupar en dos clases o tipos de formas de combinar la información de los clasificadores base. Por un lado se encuentran los métodos llamados de “Fusión” que pretenden combinar la información disponible de cada

[a]Detrás de la idea de dividir la base de datos original en otras bases existe un abanico variado de posibilidades planteadas en la literatura y permanece aún como un campo en pleno desarrollo. Entre otras estrategias se plantean la utilización de técnicas de sobre-muestreo como SMOTE [4][5] o de submuestreo, o una combinación de las mismas [48]

clasificador para una muestra dada. Es decir, cada muestra se clasifica utilizando todos los clasificadores base y en función del resultado que arrojan estos, se toma una decisión final. Existen variados métodos de combinación de clasificadores mediante fusión, y estos varían según la naturaleza de los clasificadores que estamos combinando. A modo de ejemplo, las posibilidades de fusión de los clasificadores son más amplias si cada clasificador tiene como salida una medida de la probabilidad de pertenecer a una u otra clase. Distinto es el caso de clasificadores como vecino más cercano u OPF donde solo se tiene una etiqueta que corresponde a la salida pero no se dispone de información que indique la confianza que se tiene sobre esa etiqueta. Un modo de combinar clasificadores mediante fusión es el voto por mayoría o el voto por mayoría ponderada. Más adelante en este capítulo se describirán en detalle algunos de los métodos de fusión más utilizados y estudiados en la literatura.

Otra forma de combinar clasificadores es la "Combinación por Selección", de este modo en lugar de clasificar cada muestra con todos los clasificadores base y luego combinar esta información de alguna manera, para cada muestra se selecciona un único clasificador del conjunto base. ¿Cómo se selecciona qué clasificador utilizar para cada muestra? Mediante entrenamiento, se observa qué clasificador presenta mejor *desempeño* en cada parte del espacio, y se asigna a cada porción del espacio un clasificador que será el responsable de tomar las decisiones en esa zona. Por supuesto que establecer qué clasificador actuará en cada región del espacio se realiza mediante entrenamiento utilizando una base para tales efectos. También es necesario establecer cuál es el criterio que define el desempeño del clasificador, como se mencionó en capítulos anteriores, dependiendo del problema que estamos atacando el porcentaje de error puede ser una buena medida de performance mientras que en problemas con gran desbalance de clases no son adecuadas este tipo de medidas y debemos recurrir a otras estrategias como por ejemplo medir el F_{value}, $recall$ o $presicion$.

Existen otros métodos de combinación de clasificadores que desde el punto de vista formal no se pueden enmarcar en las categorías establecidas anteriormente, sin embargo la gran mayoría de los métodos son combinación de las dos estrategias antes establecidas. Otros métodos de combinación de clasificadores que se pueden encontrar utilizan los clasificadores en cascada o mezclan los conceptos de fusión y selección. Supongamos por ejemplo que se desea combinar un clasificador SVM con uno de vecino más cercano, supongamos además (para simplificar la ilustración del ejemplo) que el problema presenta únicamente dos clases, un modo de combinación utilizado es ponderar la etiqueta dada por SVM mediante una función de la distancia a la frontera de decisión, de este modo los patrones que están más cerca del borde de las clases serán clasificados (fundamentalmente) por el valor que tome el criterio del clasificador de vecino más cercano, mientras que para las muestras que caigan suficientemente lejos del borde de decisión poco importara el valor que asigne a esa muestra el clasificador de vecino más cercano.

7.1.1. Fusión

Como se mencionó anteriormente, existen varias maneras de combinar salidas de distintos clasificadores, además el modo de combinar las mismas está vinculado en muchas circunstancias a la naturaleza de las salidas que arrojen los clasificadores base. Xu et al.[21] hace distinción entre 3 tipos de salidas para los clasificadores:

- Primer Tipo:
  Cada clasificador D_i produce una etiqueta que corresponde a una de las clases posibles $s_i \in \Omega$ y supongamos que disponemos de L clasificadores, entonces $i = 1, ..., L$ y para cada muestra obtenemos $s = [s_1 \ ... \ s_L] \in \Omega^L$. Para este tipo de salidas, no se posee información de la confianza que se tiene sobre cada etiqueta o una idea de cuales clases son las mejores candidatas para dicha muestra.

- Segundo Tipo:
  Otro tipo de salidas que podemos tener en un clasificador, en lugar de una etiqueta que vincula a cada muestra con una de las clases del problema ($s_i \in \Omega$) consiste en un conjunto de etiquetas para cada muestra con las clases más probables. Es decir que la salida para cada D_i es un subconjunto (ordenado o no) de Ω. Este tipo de salidas es muy común en problemas con un número elevado de clases como por ejemplo reconocimiento facial, de caracteres o reconocimiento de voz. En el problema que se aborda en este trabajo, donde solo tenemos dos clases (consumos anómalos y normales) este tipo de salidas no son frecuentes (ni útiles).

- Tercer tipo:
  El último tipo de salidas que se considerará se refiere a aquellos clasificadores cuyas salidas son un vector de igual dimensión que el número de clases, con componentes continuas. Supongamos que tenemos c clases, entonces la salida en este caso será de la forma $s_i = [x_1^i \ ... \ x_c^i]$ donde se puede suponer sin perder generalidad que $x_p \in [0, 1]$ para $p = 1...c$. El valor de x_p^i representa la probabilidad dada por el clasificador D_i que la muestra pertenezca a la clase ω_p.

Voto por mayoría

El método de combinación de clasificadores mediante voto por mayoría, es una de las estrategias más antiguas y sencillas a la hora de combinar clasificadores. La idea es simple y no requiere de ningún tipo de salida especial, basta únicamente con la etiqueta que asigna cada clasificador. Asumamos que se tiene un problema de c clases y la salida de cada clasificador la podemos describir mediante un vector binario de dimensión c, que toma valor 1 para la clase seleccionada. Tenemos entonces, $s = [d_1^i \ ... \ d_c^i] \in \{0, \ 1\}^c$ para $i \in L$ (L corresponde al número de clasificadores). $d_p^i = 1$ si el clasificador D_i asigna la muestra a la clase p y cero en caso contrario. El conjunto de clasificadores asignará a la etiqueta correspondiente a la clase ω_k si:

$$\sum_{i=1}^{i=L} d_k^i = max_{p=1..c} \left\{ \sum_{i=1}^{i=L} d_p^i \right\} \tag{7.1}$$

En otras palabras a cada muestra se asignará la clase más votada dentro del conjunto de clasificadores en consideración.

Voto por mayoría ponderada

Una vez introducido el concepto de voto por mayoría, lo primero que puede venir a la mente es ¿Qué ocurre si tenemos clasificadores con *desempeños* dispares? ¿Es razonable dar el mismo valor al peso de uno que al de otro? Como respuesta a estas preguntas surge la idea de modificar el método antes introducido asignando pesos a las etiquetas que da cada clasificador base. La idea es que los pesos sean mayores cuanto más acertado es el clasificador. Nuevamente se cae en el problema de definir adecuadamente **criterios que permitan evaluar la performance** de los clasificadores para determinar cuál es más preciso y cuál comete mayores errores, recordemos una vez más, que en problemas donde las clases son desbalanceadas se debe tener especial cuidado en este aspecto.

Se asume al igual que en el caso anterior se tienen L clasificadores y c clases posibles, donde $d_p^i(x) = 1$ si el clasificador i asigna a la muestra x a la clase p y cero en cualquier otro caso. Se define para cada clase p la función $g_p(x)$ dada por

$$g_p(x) = \sum_{i=1}^{L} b_i \, d_p^i(x) \tag{7.2}$$

donde b_i es el peso que se asignara al clasificador i. De este modo la función $g_p(x)$ será la suma de los pesos de los clasificadores que para x asignaron la clase ω_p.

El conjunto de clasificadores asignara para cada muestra x la clase que maximiza la expresión dada en (7.2).

Para terminar de definir este criterio de combinación, aún falta determinar como se asignarán los pesos b_i para cada clasificador y como se debe alterar la ecuación (7.2) si las clases no son equiprobables. Kuncheva determina bajo ciertas hipótesis[b] en [23] los pesos b_i que maximizan la probabilidad de acierto del conjunto de clasificadores en función de la probabilidad de acierto individual de cada uno de los clasificadores base que estamos combinando. Estos vienen dados por[c]:

$$b_i \propto log\left(\frac{p_i}{1-p_i}\right) \tag{7.3}$$

Finalmente se obtiene $g_p(x)$ como:

$$g_p(x) = log\left(P(\omega_p)\right) + \sum_{i=1}^{L} d_p^i \, log\left(\frac{p_i}{1-p_1}\right) \tag{7.4}$$

El primer término de la ecuación (7.4) corresponde al logaritmo de la probabilidad a priori de que una muestra pertenezca a la clase ω_p.

Los coeficientes b_i son aquellos que lograrán mayor probabilidad de acierto en las condiciones de independencia tomadas como hipótesis. Antes de considerar este

[b] L clasificadores **independientes**, con probabilidades de acierto $p_1 \cdots p_L$

[c] la demostración se puede consultar en [23, pag.125]

criterio como un buen método de combinación de clasificadores, es necesario hacer reparo en lo siguiente; supongamos que tenemos tres clasificadores con probabilidades de acierto p_1 , p_2 y p_3, utilizando los coeficientes dados en (7.3) obtendremos de la combinación de estos clasificadores un nuevo clasificador (resultante de la combinación de los tres clasificadores base) con probabilidad de acierto p_4^{maj} que es la máxima probabilidad de acierto que podemos obtener al variar los coeficientes $b_1,\ b_2$ y b_3. Ahora bien, supongamos que la probabilidad de acierto $p_4^{maj} > max\{p_1,\ p_2,\ p_3\}$ aún en estas condiciones ¿Hemos avanzado en la búsqueda de un mejor clasificador para el problema que se pretende abordar? En la mayoría de los problemas la respuesta es, sí!. Sin embargo en problemas donde las clases presentan grandes desbalances la respuesta no es tan sencilla. Supongamos para fijar ideas que estamos atacando un problema donde las muestras de la clase minoritaria representan el $1\,\%$ de las muestras totales, observemos que ocurre en el caso en que D_1 y D_2 clasifican **todos** los patrones como de la clase mayoritaria sin importar el valor que tomen las características y el tercer clasificador (D_3) clasifica la mitad de las muestras de la clase minoritaria correctamente pero comete errores al clasificar algunas de las muestras de la clase mayoritaria. En ese caso tendremos $p_1 = p_2 = 99\,\%$ y asumamos $p_3 = 70\,\%$.

¿Qué resultará de la combinación de los clasificadores mediante el método antes descripto? Si se calculan los pesos, es fácil observar que el clasificador D_3 tendrá poca incidencia en la decisiones y el clasificador resultante de la combinación tendrá un comportamiento similar al de D_1 y D_2 . Si bien el porcentaje de aciertos será elevado (ya que se cometen errores únicamente al clasificar muestras de la clase minoritaria) el clasificador resultante de la combinación es muy inferior a D_3 si lo comparamos en términos, por ejemplo, del F_{value}. En otras palabras, se debe prestar particular atención a los métodos que permiten mejorar el porcentaje de acierto cuando una de las clases (y en este caso la más importante) es mucho menos probable que la clase mayoritaria.

Para sortear el problema que se plantea en el párrafo anterior, en los capítulos correspondientes a la implementación de la solución a la detección de consumos anómalos se presentarán otras alternativas que fueron implementadas y evaluadas para asignar pesos a los clasificadores teniendo en cuenta el desbalance de las clases.

Por otro lado, los esquemas que combinan clasificadores atendiendo únicamente al desempeño individual de cada uno de ellos presentan otras carencias como, por ejemplo, ignorar la diversidad de los clasificadores. Supongamos para fijar ideas que tenemos nuevamente que combinar 3 clasificadores D_1, D_2 y D_3, donde los dos primeros son casi el mismo clasificador (las etiquetas que arrojan coinciden mayoritariamente) y el tercer clasificador se complementa bien con éstos, en el entendido de que no comete errores sobre las mismas muestras que D_1 y D_2. En el esquema de voto por mayoría, el desempeño del la combinación será muy similar al de D_1 (que a si vez es similar a D_2) pues, en general D_1 y D_2 *votan juntos contra* D_3. De este modo desperdiciamos esa diversidad existente entre los clasificadores que aportan información complementaria que podría ser mejor utilizada. La diversidad en los clasificadores es parte fundamental del éxito de la combinación [23][43].

7.1.2. Selección

La idea de utilizar distintos clasificadores para distintas muestras se remonta a 1978, cuando Dasarathy y Sheela [8] proponen combinar un clasificador lineal con uno de k-vecinos más cercanos, la idea básica era establecer una zona en el espacio de muestras en el cual el clasificador lineal presentaba un índice elevado de errores (cerca de la frontera de decisión), dentro de esa zona se utilizaba para etiquetar las muestras el clasificador de k-vecinos mientras que en el resto del espacio la decisión se tomaba apelando a la etiqueta que arroja el clasificador lineal.

Antes de implementar cualquier estrategia de combinación de clasificadores mediante selección, algunos aspectos deben ser tenidos en cuenta. Por ejemplo ¿Cómo se construirán los clasificadores base?, ¿cómo se evaluará el desempeño de cada clasificador dada una muestra x? En el caso que varios clasificadores tengan igual desempeño para un conjunto de muestras ¿Cómo se plantea el desempate? ¿Es conveniente utilizar únicamente el clasificador más competente o realizar fusión entre el conjunto de clasificadores más competentes? Luego de resolver los aspectos antes mencionados, se particiona el espacio en el que se encuentran las muestras (típicamente $\mathbb{R}^n$ para n características) en p regiones ($K_i \in \mathbb{R}^n \;\; i : 1..p$), para cada región del espacio se define el o los clasificadores que tomarán las decisiones para las muestras que caigan en dicha región. Las regiones que se obtienen luego de definir la estrategia de selección y los clasificadores que se desea combinar, no tienen en principio correlación alguna con las clases del problema. Es decir, no hay clases asociadas a estas regiones, únicamente se asocian clasificadores responsables de tomar las decisiones en cada región.

Una ventaja que presentan los métodos de selección con respecto a los de fusión, consiste en que los métodos de selección, por diseño aseguran por lo menos la performance del mejor de los clasificadores que estamos combinando. Sin embargo los métodos de selección pueden presentar problemas de sobreajuste a los datos de entrenamiento. Para evitar estos problemas se suele asignar un único clasificador a una región sólo cuando éste presenta un desempeño considerablemente superior al resto, de lo contrario puede ser mejor estrategia implementar métodos híbridos en los cuales para cada región, fusionamos de determinada manera los mejores clasificadores del conjunto original.

7.2. Otras estrategias de combinación

En los últimos años han surgido una gran cantidad de variantes a la hora de combinar clasificadores. Recordemos que uno de los aspectos fundamentales para tener éxito a hora de la combinación de clasificadores radica en que los mismos posean cierta independencia entre sí. Como se mencionó, esto se puede lograr en varios niveles (tomando distintos datos para cada clasificador, distintas características, etc.). En este aspecto, han surgido varias técnicas que permiten lograr buenos desempeños en la combinación, aún teniendo una sola base de datos. Entre estas técnicas se encuentran *Bagging*, *Boosting* y *SmoteBoost*, entre otras. A continuación pasaremos a describir las principales técnicas utilizadas en problemas donde se posee un número limitado de datos.

7.2.1. Bagging

El término *Bagging* fue introducido por Breiman como un acrónimo de *Bootsrap AGGregatING*[31]. La idea es simple y pretende abordar problemas en los cuales no tenemos suficientes datos como para tomar una base independiente para cada uno de los clasificadores que deseamos combinar. Dada un base de datos, se realiza un muestreo con reposición, para generar p bases de datos. El tamaño de las p bases resultantes es un parámetro que se debe determinar en función del problema que se esta abordando, como norma general, se suele crear bases de datos del mismo tamaño que la base original (obviamente en las nuevas bases de datos puede haber muestras repetidas una o más veces). Por ultimo, una vez que generamos las p bases de datos, entrenamos los p clasificadores y combinamos las salidas mediantes alguno de los esquemas de combinación descritos en las secciones anteriores.

Si bien usualmente este método se presenta como un algoritmo de combinación de clasificadores, muchas veces se utiliza como un método de estabilizar clasificadores inestables (como árboles o redes neuronales). Por ejemplo, supongamos que queremos entrenar un clasificador de tipo árbol que presenta grandes variaciones respecto a entrenamientos sucesivos. En este caso podemos utilizar la técnica de bagging para crear **un** clasificador, mediante la combinación del mismo en varias instancias como se explicará a continuación. Mediante muestreo con repetición de la base de datos que tenemos para el entrenamiento, creamos p bases de datos (en general de igual tamaño que la base original), luego corremos p instancias del árbol (entrenando en cada instancia con una de las p bases). De este modo obtenemos p árboles **distintos**[d]. Luego combinamos[e] los p árboles para obtener como resultado **un** clasificador que es más estable que cada una de las instancias individuales.

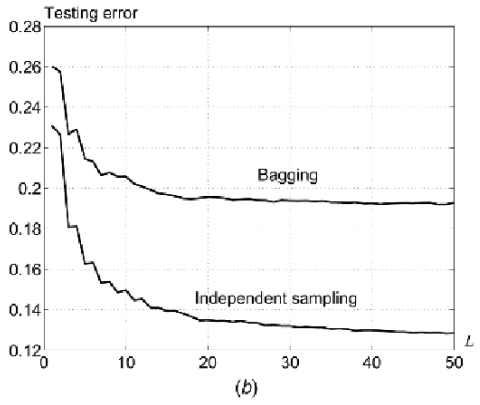


Figura 7.1: Error sobre una base de entrenamiento en función del número de instancias que se combinan de un mismo clasificador. Se comparan los resultados utilizando bagging para generar las distintas bases de entrenamiento con lo obtenido utilizando conjuntos independientes. Imagen tomada de [23]

[d] aclaremos que los árboles no son **independientes** pues no estamos utilizando bases independientes para cada instancia

[e] utilizando fusión de las salidas o algún otro método de combinación

En la figura 7.1 se puede observar como varía el error sobre una base de test luego de aplicar bagging para combinar varias instancias de un mismo clasificador. Como se puede observar los resultados son peores que los obtenidos si se utilizan p bases independientes, sin embargo en muchos casos no se poseen datos suficientes como para tomar bases independientes para entrenar cada instancia.

7.2.2. Boosting

Boosting es un algoritmo inspirado en $Hedge(\beta)$[15], el cual asume que tenemos un conjunto de estrategias para predecir un determinado fenómeno. Estas estrategias son ponderadas por un coeficiente que se va modificando en función del nivel de acierto o error que van teniendo cada una de las estrategias posibles. El algoritmo evoluciona en función de la performance de las decisiones que se van tomando y tiene la propiedad que el sistema tiende a mejorar su desempeño aún cuando las estrategias que se están combinando permanezcan siendo pobres. En [23, pag. 213] se puede consultar el algoritmo en más detalle aplicado al caso de reconocimiento de patrones.

AdaBoost

El algoritmo Adaboost (por ADAptive BOOSTing) adopta la idea detrás de $Hedge(\beta)$ y la aplica al caso de combinación de clasificadores, dando una estrategia de combinación de clasificadores que pretende desembocar en una regla de decisión precisa partiendo de la combinación de clasificadores con una moderada o mala performance.

Asumamos que tenemos que combinar m clasificadores (D_1 ... D_m) y tenemos una base de datos B para realizar los entrenamientos. Para entrenar cada clasificador se hará un muestreo con reposición de la base B con el siguiente criterio, como condición inicial comenzamos muestreando elementos de B donde la probabilidad de elegir cada elemento es la misma, es decir probabilidad uniforme sobre los elementos de B. Con el subconjunto obtenido de B, llamemoslo B_1, entrenamos el clasificador D_1, cometiendo error en la clasificación sobre los elementos $x_1,\ ...\ x_n \in B_1$. A los elementos $x_1,\ ...\ x_n \in B_1$ se les incrementa la probabilidad de ser escogidos dentro de B en función de un cierto peso. De este modo cuando se muestrea B para obtener B_2 (base que utilizaremos para entrenar D_2) existe más probabilidad de escoger $x_1,\ ...\ x_n$ que el resto de los elementos. Se entrena D_2, se observan los elementos para los cuales se comete error, se actualiza el peso de escoger los mismos y así sucesivamente hasta obtener todo el conjunto de clasificadores entrenados. De este modo a medida que se corre el algoritmo, los elementos de la base de datos que se está utilizando, sobre los cuales se cometen errores, cobran mayor importancia que los elementos que son correctamente clasificados. Cuando finaliza el algoritmo se obtiene los m clasificadores entrenados y m coeficientes $\beta_1,\ ...\ \beta_n$ calculados en función de los errores que cometió cada clasificador en la etapa de entrenamiento[f].

[f] se puede consultar en detalle el algoritmo en [23, pag.216]

Finalizado el proceso anterior, definimos el soporte de cada clase ω_t como:

$$\mu_t(x) = \sum_{D_k(x)=\omega_k} ln\left(\frac{1}{\beta_k}\right) \tag{7.5}$$

La clase con mayor soporte sera la que se asigne a la muestra x.

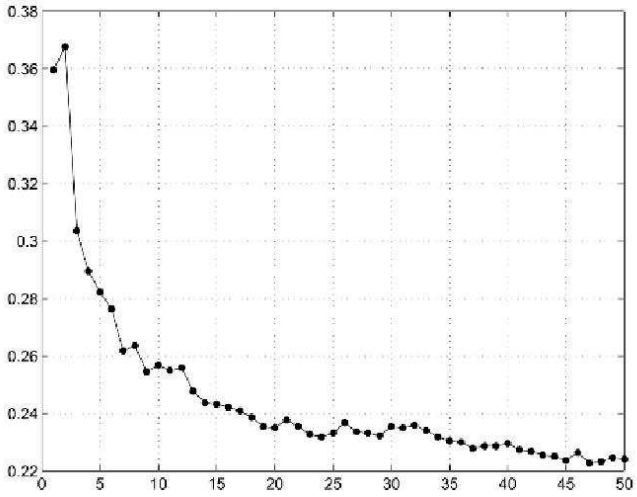


Figura 7.2: Error en Test en función de la cantidad de instancias del clasificador que se utilizan. Imagen tomada de [23]

Al igual que en el caso de bagging, si bien en principio AdaBoost se plantea como una estrategia de combinación de clasificadores, en muchos casos se utiliza como manera de estabilizar clasificadores que presentan variaciones o como método de mejora de clasificadores que considerados individualmente tienen pobre performance. En estos casos lo que se hace es combinar el mismo clasificador con distintas instancias del mismo, en cada etapa de entrenamiento se toma mayor consideración de las muestras que fueron clasificadas erróneamente en etapas anteriores. De este modo se combinan varias instancias de un mismo clasificador obteniendo un nuevo "clasificador" con mejor desempeño y estabilidad que si se utiliza una única instancia entrenada con toda la base.

Variantes para el caso particular de desbalance de clases

En problemas en los cuales se tienen muchas más muestras de la clase mayoritaria que de la clase minoritaria, se plantean algunos métodos particulares que aprovechar al máximo las muestras de la clase minoritaria.

Una estrategia que se propone en [1] consiste en utilizar para el entrenamiento de cada uno de los clasificadores base, **todas** las muestras de la clase minoritaria y un muestreo (sin reposición) de los elementos de la clase mayoritaria. De este modo todos los clasificadores son entrenados con todos los elementos de la clase minoritaria pero distintos elementos de la clase mayoritaria (obteniendo cierta independencia y aprovechando las muestras minoritarias a la vez). Luego las salidas de cada clasificador se combinan

mediante algún método tradicional de fusión o selección.

Otro método propuesto por Barandela et al. [34] similar al anterior, consiste en utilizar para entrenar cada clasificador base todos los elementos de la clase minoritaria, y un muestreo de la misma cantidad de elementos de la clase mayoritaria. De este modo cada clasificador base es entrenado con todos los elementos de la clase minoritaria y algunos elementos de la clase mayoritaria. A diferencia del caso anterior, en el algoritmo propuesto por Barandela et al. todos los clasificadores se entrenan con bases balanceadas. Esto puede ser favorable (o no) dependiendo del clasificador que se va a utilizar y del modo en que se ajustan los parámetros del mismo. En los capítulos en que se describe la implementación de la solución se entrará en mayor detalle en este aspecto.

Por último mencionemos otro método que está siendo utilizado en la actualidad, llamado *SMOTEBoost*[5], que consiste en aplicar el método de Boosting antes mencionado pero utilizando previamente el algoritmo de SMOTE para generar muestras sintéticas de la clase minoritaria.

Parte III

Implementación de la solución

CAPÍTULO 8

Bases de datos y su pre-procesamiento

Antes de desarrollar la herramienta para clasificar los patrones es importante analizar la *entrada* a nuestro sistema, los consumos mensuales, sus etiquetas y otras variables de interés. En este proyecto se utilizaron 3 bases de datos brindadas por UTE:

1. **Base 1**: utilizada en el trabajo previo de Kosut y Alcetegaray [10]. Son alrededor de 500 datos que corresponden a una porción especial de autoservicios. Esta base fue utilizada por ellos para probar la viabilidad de utilizar técnicas de reconocimiento de patrones para clasificar suministros, por lo que se eligieron (manualmente) suministros con un comportamiento tal que logren caracterizar los más posible las dos clases, consumos **anómalos** y **normales**, y así lograr una buena performance.

2. **Base 2**: 1504 suministros industriales de distinta índole (almacén, supermercado, autoservicio y almacén/vivienda) seleccionados de forma aleatoria por los técnicos de UTE, a la cual no se le realizó ninguna pre-selección previa buscando un comportamiento lo más real posible de los mismos.

3. **Base 3**: 3338 suministros industriales (los cerca de 1500 de la base anterior están entre estos suministros) con facturación hasta Enero de 2011.

Al comenzar el proyecto se contaba con las bases 1 y 2, sin embargo solo la base 2 es realmente adecuada para desarrollar los distintos métodos de clasificación, probar su viabilidad, analizar su performance y generar la herramienta final de clasificación ya que está formada por un numero de suministros suficientes para caracterizar a ambas clases (aunque existe un gran desbalance de clases) y porque fueron elegidos de forma aleatoria, a diferencia de la base 1. Durante el proceso de desarrollo de los métodos de clasificación, se utiliza la base 1 para tener una medida de éxito de los mismos comparando con los resultados obtenidos en [10]. Finalmente, la base 3 es utilizada para evaluar el éxito de la herramienta final de clasificación a través de inspecciones realizadas por los técnicos de UTE a partir de las etiquetas asignadas por la herramienta.

8.1. Composición de las Bases de Datos

Las bases de datos están compuestas por suministros tomados por los técnicos de UTE correspondientes a la energía total consumida por cada cliente en un intervalo aproximado de un mes. En particular, la base 2 contiene consumos de clientes empezando en Octubre de 2004 y terminando en Setiembre de 2009, mientras que la base 3 contiene consumos empezando en Enero de 2008 y terminando en Enero de 2011. Para la base 1 no se tiene la información de fechas a las cuales corresponden las medidas de consumo.

Aparte del consumo mensual de cada cliente las bases 1 y 2 cuentan con una información imprescindible para nosotros, las etiquetas (en la base 3 no se cuenta con las etiquetas porque no se necesitan ya que se utiliza únicamente para evaluar la performance del clasificador final comparando con el resultado de las inspecciones). En los problemas de reconocimiento de patrones son muy importantes ya que dicen a qué clase (en nuestro caso, anómalo o normal) pertenece cada suministro, lo cual se utiliza para entrenar los distintos clasificadores y evaluar su performance. Las etiquetas son datos externos al problema, suministrados por expertos en el tema. En nuestro caso fueron los ingenieros de UTE quienes analizaron manualmente los suministros y los etiquetaron en base a experiencia acumulada y a criterios que se establecen en la empresa que caracterizan a un posible cliente fraudulento. Esta forma de etiquetar presenta un problema ya que a pesar de ser realizada por expertos en el tema, la decisión es sumamente subjetiva y para nada contundente en ciertos casos. Además, puede ser que el etiquetado haya sido realizado teniendo en cuenta factores más allá de los consumos. Esto sin duda puede confundir al clasificador a la hora de ser entrenado en las propiedades de cada clase.

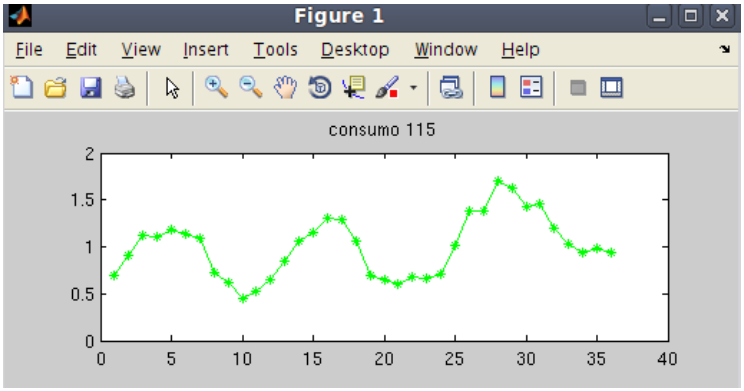
En la figura 8.1 se muestra un ejemplo de los 2 tipos de consumos dentro de la base de datos. Por un lado el consumo que se considera normal, presentando una forma de consumo que se repite en los 3 años y por otro, un consumo que es sospechoso de ser fraudulento, lo que se demuestra en este caso por su caída constante del consumo.

La fecha de detección de irregularidades es otra variable que proporciona información valiosa al utilizar estos suministros para entrenar los clasificadores. Esta información proviene de inspecciones realizadas por UTE, las cuales se puede deber a un etiqueta previa o simplemente por rutina.

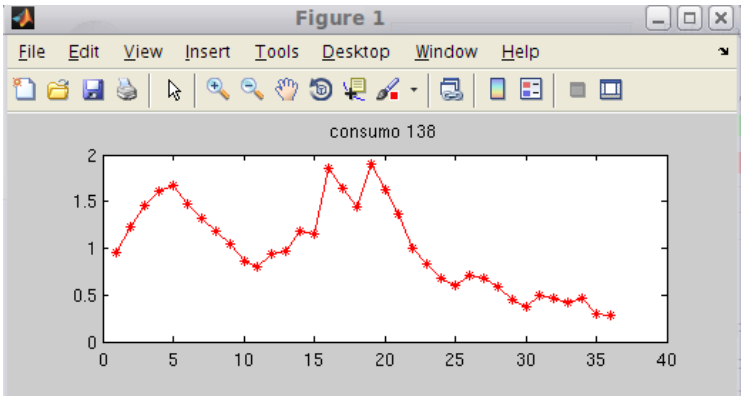
Además de la información descrita anteriormente, la base 3 cuenta con ID del suministro, CNAE (rótulo que caracteriza a los clientes en función del rubro y naturaleza de consumo) y una variable diciendo si es un suministro el cual va a ser inspeccionado por UTE en un futuro.

8.2. Pre-procesamiento

La determinación de la etiqueta que se le asignará a un determinado cliente, normal o sospechoso, con el proceso de reconocimiento de patrones está ligado al análisis entre las características de un mismo cliente, es decir que el estudio realizado está concentrado en el cliente en sí mismo y no comparando todos los clientes o tratando de



(a)



(b)

Figura 8.1: Ejemplo de 2 consumos pertenecientes a cada una de las clases de nuestro problema. La figura (a) es un consumo etiquetado como normal, mientras que la curva mostrada en (b) representa a un consumo etiquetado como anómalo

determinar un consumo de cliente característico para luego comparar con cada cliente. Es por esta razón que se debe considerar una determinada cantidad de medidas de consumo fija para cada cliente. La decisión de cuántas medidas de consumo tomar está relacionada con cuál es el mínimo necesario y cuál es un máximo eficiente, es decir a partir de cuántas medidas ya no aportan información útil. Tomar consumos muy antiguos puede no ser relevante, el interés en el análisis está dado en la comparación de los últimos consumos mensuales de un cliente (comportamiento actual) con el consumo característico o típico de ese cliente ya que se asume que el fraude se comete durante un determinado período de tiempo y no permanentemente, y únicamente interesa detectar los fraudes actuales. Demasiados consumos mensuales no solo aumentarían el costo computacional del problema si no que también puede hacer que muchos suministros no cumplan con esa cantidad de consumos y por lo tanto queden eliminados. Causas de esto pueden ser cambios de dueño, periodos sin consumo o clientes nuevos. Entendiendo los factores involucrados en la determinación de la etiqueta y analizando este caso en particular con los técnicos de UTE es que se determinó que *la cantidad óptima de medidas a considerar*

por cada cliente es 36.

El problema de **detección de consumos anómalos** presenta determinadas características que hacen fundamental tener una primera instancia de **pre-procesamiento** de los datos:

- Una característica determinante es el desbalance de clases, tanto en la diferencia entre la cantidad de consumidores que "roban" energía y los que no, como entre las muestras de clientes "sospechosos" y "normales" con las que se cuenta. En el capítulo 4 se hace referencia al problema del desbalance de clases introduciendo diferentes técnicas de aproximación al mismo, ya que las tradicionales han demostrado ser poco efectivas.

- La forma en que se toman las medidas de cada cliente presenta un problema. Los técnicos de UTE encargados de las mediciones en teoría visitan el cliente una vez por mes (idealmente cada 30 días). Sin embargo esto no es realmente así, por lo que al no realizar las medidas cada una cantidad fija de días, los datos de las mediciones presentan un cierto ruido. Por ejemplo, si un cliente consume exactamente lo mismo todos los meses, dos medidas consecutivas pueden no mostrarlo si no se realizan ambos cada una cantidad igual de días. Otro problema importante es que en ciertas ocasiones no se puede tomar la medida de un determinado cliente por no poder acceder al contador (por ejemplo cuando el cliente no se encuentra en su hogar). En estos casos UTE estima el consumo a partir del histórico y luego lo corrige cuando se logra acceder al consumo real. Estas dos consideraciones hacen que las medidas de consumo de los usuarios puedan no representar de forma fidedigna el comportamiento de los mismos.

Procurando disminuir el desbalance de clases, en la etapa de pre-procesamiento se decidió utilizar la etiqueta de fecha de detección de irregularidades. Los suministros que presentaron una detección de irregularidades fueron re-etiquetados como anómalos y se cortaron para tomar los 36 meses antes de la fecha de detección de la irregularidad. Esto presenta un problema, se tienen registros de las irregularidades detectadas pero eso no quiere decir que el resto de los consumos no estén cometiendo fraude. Es seguro que algunos de los suministros de los que se asume por defecto como normales, podrían a priori ser suministros fraudulentos a los que simplemente aún no se ha inspeccionado o se los ha inspeccionado pero por la naturaleza del ilícito no se ha podido detectar durante la inspección. Este etiquetado, junto con el manual hecho por los técnicos de UTE, tienen sus particularidades, pero partiremos de la hipótesis de que incluir las 2 formas de etiquetar anómalos es la mejor solución al problema.

El inconveniente introducido unos párrafos atrás por la forma en que se toman las medidas sumado a la estimación de consumos por no poder acceder a ellos con su posterior corrección hace que los suministros presenten picos de consumo o variaciones, no reales. Para solucionar esto, durante el pre-procesamiento se **filtran** las medidas de consumo de los suministros sacando los picos que son sospechosos de no corresponder con la realidad.

Finalmente, suministros que presentan varios consumos mensuales iguales a 0 (se puede deber a un cambio de firma) son considerados suministros ruidosos, por lo que

son eliminados en esta etapa.

8.3. Nomenclatura

A lo largo de este documento se hará referencia a distintas implementaciones de clasificadores, distintos conjuntos de características propuestos y ensayados así como también se trabajará con diferentes bases de datos. A continuación se presenta un resumen de los términos que se utilizarán indicando a quien corresponden.

Bases de Datos

- Base-1
  archivo: `DeCA/Bases de dato/cons_etiq_pru2.csv`
  Base de datos correspondiente a suministros de autoservicios en Montevideo (500 clientes aprox).

- Base-2
  archivo: `DeCA/Bases de dato/listadoConFecha_v2.csv` Base de datos que contiene autoservicios, panaderias, minimercados y carnicerias (1500 clientes aprox).

- Base-3
  archivo: `DeCA/Bases de dato/base_3.csv` Base de datos que contiene locales comerciales similares a los de la Base-2 (3000 clientes aprox).

Características

- Car-A
  Conjunto de características propuestas en [10].

- Car-DeCA
  Conjunto de características implementadas:
  Archivo:`DeCA/v.1.0.0/crea_caracteristicas_v4.m`

- Car-C
  Se toman como características el valor del consumo promedio mensual, es decir el propio vector de datos suministrado por UTE.

Clasificadores

- One Class SVM
  Archivo:`DeCA/v.1.0.0/clasificador_svmoneclas_en_C(_train/_test).m`

- WC-SVM o CS-SVM
  Archivo:`DeCA/v.1.0.0/clasificador_libsvm(_train/_test).m`

- OPF
  Archivo:`DeCA/v.1.0.0/OPF(_train/_test)_sl.m`

- Tree (C4.5-Adaboost)
  Archivo:`DeCA/v.1.0.0/clasificador_tree_prtools(_train/_test).m`

CAPÍTULO 9

Características

El éxito en todo proceso de reconocimiento de patrones depende, en buena medida, de qué tan bien se entienda el problema. Es vital intentar desarrollar el conocimiento y la habilidad que posee el experto, y que le permite clasificar los consumos al observarlos. Ubicarse en el trabajo de los expertos y entrar en el contexto en el que realizan la clasificación, tratar de entender qué variables entran en juego, qué información es tenida en cuenta (de manera consciente e inconsciente) y cómo se combinan para desembocar en una decisión.

A lo largo de este capítulo, se describirán brevemente las características que se utilizarán. También se tratará de transmitir los motivos fundamentales que inspiraron la consideración de cada característica y qué tipos de fraude pretenden contemplar.

En una primera aproximación al problema, se consultó cómo se había abordado la temática en otras partes del mundo, los primeros artículos consultados en esta etapa fueron [36] [10] [29] y [30]. No se encontró en la bibliografía consultada consenso desde el punto de vista de las características que se deben utilizar. Además, las mismas están atadas en buena medida a las metodologías de cada compañía eléctrica. En este caso de estudio, no se cuenta con los recursos disponibles para obtener algunas de las características utilizadas en los trabajos citados, motivo por el cual, se tuvieron que plantear características nuevas, inspiradas en algunos casos en trabajos previos y en otros casos inferidas a partir de la información provista en las reuniones con los técnicos.

9.1. Descripción de las características

Cociente entre los valores medios

Una de las formas en que se puede manifestar un consumo fraudulento, es mediante un cambio en el valor medio del consumo. Lógicamente cuando se efectúa un fraude, la intención es beneficiarse mediante una reducción en los pagos mensuales. Para

lograr ese “ahorro” en las facturas, procurará por distintos medios, disminuir el valor medio de su consumo mensual.

Como primeras tres características, se utilizará la comparación de la media en los últimos 3, 6 y 12 meses con la media anterior del consumo [a]. Es decir:

$$car1 = \frac{mean(C[n-3:n])}{mean(C[1:n-4])} \tag{9.1}$$

$$car2 = \frac{mean(C[n-6:n])}{mean(C[1:n-7])} \tag{9.2}$$

$$car2 = \frac{mean(C[n-12:n])}{mean(C[1:n-13])} \tag{9.3}$$

donde n es el número de meses considerados en el consumo.

Norma de la diferencia entre el consumo esperado y el consumo real

Por medio de esta característica se pretende observar cambios en el comportamiento del usuario. La idea es comparar cada uno de los valores de consumo de cada mes del último año, con el valor del mismo mes del año anterior. La comparación se corrige con un factor correspondiente al cociente entre las medias para independizar las medidas del valor medio. El factor de corrección pretende eliminar variaciones debidas a cambios en la temperatura media en uno y otro año, que pueden trasladar verticalmente los consumos. Finalmente sumamos el error cometido durante los últimos 12 meses del consumo.

$$car4 = \sqrt{\sum_{i=n-11}^{i=n} \left(C(i) - \alpha C(i-12)\right)^2} \tag{9.4}$$

Donde n es el número total de meses que consideramos para los consumos y α corresponde al cociente entre el consumo medio del último año y el consumo medio en el penúltimo año.

Si bien se pierde algo de información al no considerar individualmente el error cometido mes a mes se disminuye significativamente el número de características.

Diferencia en los espectros

Otro de los aspectos importantes a la hora de distinguir entre consumos, puede ser cómo es el espectro de los mismos. Es decir ¿Qué componentes de frecuencia predominan? ¿Existe alguna periodicidad? ¿Se presentan cambios en el valor de continua?. Estas y otras preguntas se pueden responder simplemente mirando en el espacio de frecuencia los distintos consumos. Como siguientes características, se propone

[a] Notación: utilizaremos $C(i)$ para referirnos al valor de un cierto consumo en el mes i

considerar la diferencia entre los coeficientes de Fourier de la curva de consumos del último año, con la curva de los años anteriores.

$$car5 = \|FFT\{C_{actual}\}(1) - FFT\{C_{medio}\}(1)\| \tag{9.5}$$
$$car6 = \|FFT\{C_{actual}\}(2) - FFT\{C_{medio}\}(2)\| \tag{9.6}$$
$$car7 = \|FFT\{C_{actual}\}(3) - FFT\{C_{medio}\}(3)\| \tag{9.7}$$

C_{actual} representa el vector de consumos durante el último año, y C_{medio} el vector promedio del consumo en los años anteriores.

Diferencia en los coeficientes wavelet

Otra manera de ver el comportamiento de una señal, es a través de otra transformación llamada Wavelets. Al igual que la Transformada de Fourier (o su versión discreta DFT) la idea es representar la señal en un espacio distinto al del tiempo en el cual se pueden resaltar, desde otro punto de vista, distintas cualidades que en el espacio del tiempo quedan ocultas o pasan desapercibidas. Al igual que en otras transformaciones (como puede ser la Transformada de Fourier), la idea es proyectar la señal original en un espacio nuevo. En el caso de la transformada de wavelets, se utiliza una función $\psi(x)$ con determinadas características[b] llamada onda madre, y la base $\{\psi_{ij}(x)\}$ del espacio en que se va a proyectar se forma por medio de compresiones y dilataciones de la onda madre además de corrimientos $\mathcal{B} = \{\psi_{ij}(x) = \psi(i\,x - j)$ con $i, j \in \mathbb{R}\}$. Esta transformación es muy usada en algunas de las referencias consultadas (por ejemplo [36]). La utilización de wavelets parece razonable por la buena capacidad que tiene este tipo de transformación para distinguir cambios abruptos y acotados en el tiempo además de permitir un análisis de una señal a distintas escalas temporales de manera simultanea.

Se utilizará como herramienta el "Wavelet Toolbox" de Matlab, que implementa la descomposición de la señal mediante el uso de bancos de filtros[c] (que dependen de la onda madre que se desee utilizar). Estos descomponen la señal obteniendo a la salida de cada filtro una porción de la señal asociada a una escala dada. En la figura 9.1 se ilustra a modo de ejemplo como se utiliza esta herramienta si se pretende estudiar una señal con un solo paso de descomposición. En la figura 9.2 se ilustra el proceso general cuando se descompone una señal utilizando múltiples niveles.

Figura 9.1: Ilustración del proceso de descomposición con un paso. Imagen extraída del manual de usuario del "Wavelet Toolbox" de Matlab.

Utilizando la interfaz gráfica disponible en el "Wavelets Toolbox" se observaron distintas alternativas combinando los niveles de descomposición y las distintas ondas madre. Se visualizaron consumos anómalos y normales seleccionados aleatoriamente y se les aplicó distintas wavelets con distintos niveles. Finalmente se decidió utilizar 2 niveles

[b] para más detalle se puede consultar [26], [37]

[c] Este método fue desarrollado por Mallar en 1988 [25] y provee un práctico algoritmo de filtrado que permite una rápida transformada de wavelet.

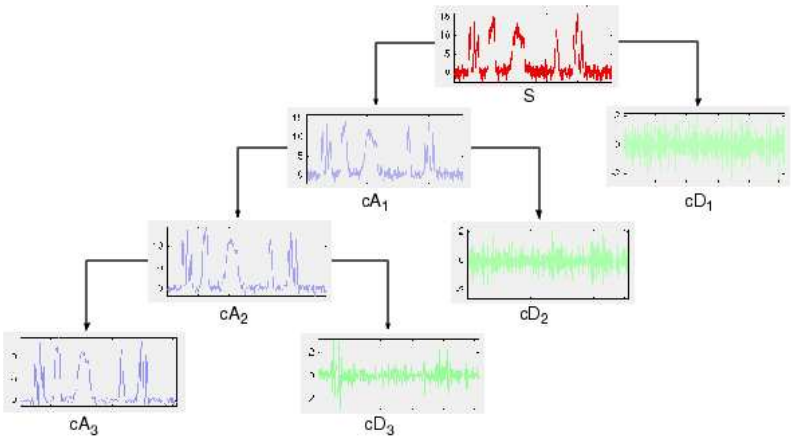


Figura 9.2: Descomposición multinivel. Imagen extraída del manual de usuario del ”Wavelet Toolbox” de Matlab.

y “db2” como onda madre.

Se llamará $cA2_u$, $cA1_u$ y $cD2_u$ a los coeficientes de ajuste en el nivel 2, 1 y de detalle en el nivel 2 respectivamente para el último año del consumo. Análogamente $cA2_p$, $cA1_p$ y $cD2_p$ para el penúltimo año del consumo.

$$car8 = |cA2_u(1)| - |cA2_p(1)| \tag{9.8}$$
$$car9 = |cA2_u(2)| - |cA2_p(2)| \tag{9.9}$$
$$car10 = |cA2_u(3)| - |cA2_p(3)| \tag{9.10}$$
$$car11 = |cA2_u(4)| - |cA2_p(4)| \tag{9.11}$$
$$car12 = |cA2_u(5)| - |cA2_p(5)| \tag{9.12}$$
$$car13 = |cD2_u - cD2_p| \tag{9.13}$$
$$car14 = |cD1_u - cD1_p| \tag{9.14}$$

Por medio de los coeficientes de ajuste se puede obtener información asociada a la tendencia del consumo relacionada con los cambios lentos o las tendencias a largo plazo, por esta razón se comparan uno a uno los coeficientes de ajuste. Por otro lado se compara el modulo de la resta de los vectores con los coeficientes de detalle, que nos permite analizar el nivel de variaciones rápidas. Al ser los coeficientes de detalle los relacionados a las variaciones más rápidas (se obtienen a la salida de los pasa-altos), al comparar estos vectores tenemos una medida de cómo está repartida la potencia de la señal a distintas escalas y a lo largo del tiempo.

Diferencia en el ajuste de un polinomio de grado N

Continuando con la caracterización de los consumos, se propone *aproximar*[d] los datos por un polinomio de grado N. Se pueden distinguir distintos casos en función del

[d] utilizando mínimos cuadrados

valor que se elija para el grado del polinomio. A modo de ejemplo, para $N = 1$ se estaría estudiando el valor medio del consumo mensual, para $N = 2$ se aproximan las curvas de consumo por una recta. Este último caso, es una de las características propuestas en [10] y que llevó a que se utilizara la aproximación por un polinomio (como generalización de ésta).

Se ajusta por medio de mínimos cuadrados, el polinomio de grado N que mejor aproxima cada año de la curva de consumos (en este caso se usará $N = 4$). Luego, se obtiene la diferencia entre el promedio de los primeros años y el último.

$$car\{15, 16, 17, 18, 19\} = polyfit(C_{ultimo\ ano}) - \frac{1}{n-1}\sum_{i=1}^{n-1} polyfit(C_{ano\ i}) \qquad (9.15)$$

Donde n es el número de años considerados

Distancia al consumo medio

Se parte de la base que existe mucho menor cantidad de consumos fraudulentos que normales. Además, se puede asumir que los consumos fraudulentos no tienen una fuerte correlación pues los tipos de fraude que se cometen son variados y se realizan en momentos independientes. Tomando dichas hipótesis, es razonable suponer que si se realiza el promedio de todos los consumos mes a mes, se obtendrá un consumo que representará en buena medida a un consumo *típico* o *normal*. Con esta idea en mente, parece razonable tomar como característica la distancia de cada consumo a dicho consumo medio, como una medida de qué tan lejos o cerca está cada consumo del consumo medio.

$$car20 = \left\| \vec{C} - \vec{C}_m \right\| \qquad (9.16)$$

Donde $\vec{C}_m$ es el consumo promedio (mes a mes) entre todos los consumos de la base de datos.

Comparación de la varianza del consumo

La demanda de energía eléctrica por parte de los usuarios presenta mayores o menores fluctuaciones dependiendo de la naturaleza del consumidor, pero en ninguno de los casos el consumo es constante. En una de las reuniones con los expertos de UTE se observó que algunos de los clientes que adulteraban el contador o realizaban *puentes* en los bornes del mismo *ajustaban* el consumo que registraba el contador. El objetivo es disminuir el monto de la factura. Para disimular el fraude, por momentos los usufructuarios conectan el contador y de manera aproximada tratan de presentar consumos parejos a los largo de los meses. Se observó que este ajuste manual hace que las curvas de consumo presenten menores fluctuaciones que los consumos normales, en los cuales el cliente no tiene un mecanismo de control del consumo. Por esta razón, consumos con varianzas muy bajas y valores muy parejos de consumo a lo largo de los meses pueden estar escondiendo fraudes del tipo descrito anteriormente.

Inspirados en las observaciones anteriores se propone considerar dos características, en primer lugar el cociente entre la varianza que presenta el consumo

promedio (año a año) con la varianza en el último período; en segundo lugar, la varianza promedio de **todos los consumos** con la varianza de cada consumo en el último año.

$$car21 = \frac{var\left(C_N\right)}{var\left(\frac{1}{N-1}\sum_{i=1}^{N-1} C_i\right)} \tag{9.17}$$

Donde C_i representa el consumo durante el año i y N el número de años considerados.

$$car22 = \frac{var\left(C_N\right)}{var\left(C_m\right)} \tag{9.18}$$

Donde C_N representa el consumo durante el último año y C_m el consumo promedio tomado entre todos los consumos.

Coeficientes de Fourier

Las siguientes 5 características, corresponden al módulo de los 5 primeros coeficientes de Fourier del consumo total. Los coeficientes de Fourier del consumo total fueron considerados en algunos trabajos previos con buenos resultados, razón por la cual se incluyeron como características,

$$car\{23, 24, 25, 26, 27\} = \left\|FFT(C)_{\{1,2,3,4,5\}}\right\| \tag{9.19}$$

Pendiente de la recta que ajusta al consumo

Como última característica, se incluye la pendiente de la recta que mejor ajusta la curva de consumos. Esta característica, resulta adecuada para obtener información de la tendencia del consumo y si el mismo presenta una caída sostenida. Además, fue considerada en trabajos anteriores [10] y parece adecuado incluirla en el conjunto final.

9.2. Otras características tenidas en cuenta

A lo largo del tiempo se fueron proponiendo numerosas características que iban surgiendo durante la maduración del problema. En cada reunión con los técnicos de UTE surgían nuevas características y se modificaba o eliminaba alguna de las existentes. Cuando se consultaban trabajos o publicaciones también se efectuaban cambios. Por último, luego de presentaciones intermedias que se realizaban del proyecto se obtuvo realimentación por parte de docentes que también llevaron a modificaciones de ciertas características o la incorporación de nuevas. A continuación se realiza un listado de algunas de las características que en algún momento fueron consideradas y que fueron eliminadas o sustituidas.

- Promedio móvil con ventanas de 3, 6 y 12 meses.
- Correlación con la temperatura. La correlación con la temperatura fue propuesta como una característica a tener en cuenta. Luego se propuso en lugar de tomar en cuenta la correlación con la temperatura de cada consumo, observar las variaciones

que presentaba dicha correlación a lo largo del tiempo, como modo de detectar el comienzo de un hecho fraudulento. Se eliminó esta característica pues en la base de datos con la que se trabajó mayoritariamente (Base-1) no todos los consumos inician en el mismo mes. Esto ocurre pues los suministros que tienen una fecha de detección de irregularidad son cortados y la irregularidad marcan el final del suministro.

- Tipo de contador (Digital o Analógico) Para la base de datos utilizada no se disponía de dicha información, si se contara con esta información sería razonable incorporar al set de características el tipo de contador.

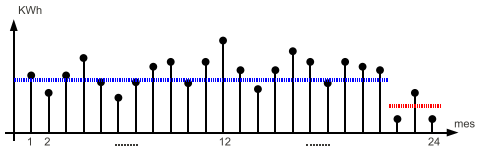
- Potencia contratada Al igual que el item anterior, no se disponía de esta información para los consumos de todas las bases de datos.

A continuación presentamos un resumen con las características finales que se proponen. Estas características luego de ser calculadas para la base de datos que se quiere utilizar, pasan por un proceso de normalización para dejarlas con media nula y varianza unitaria.

9.3. Resumen

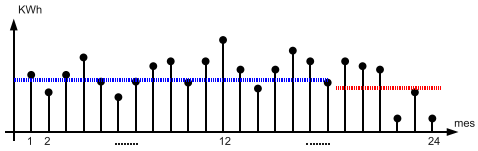
1. Cociente entre el valor medio en los últimos 3 meses y el pasado

$$car1 = \frac{mean(C[n-3:n])}{mean(C[1:n-4])} \quad (9.20)$$



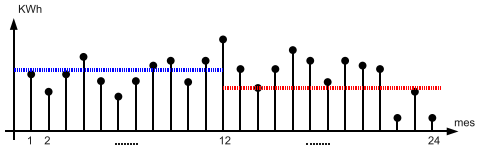
2. Cociente entre el valor medio en los últimos 6 meses y el pasado

$$car2 = \frac{mean(C[n-6:n])}{mean(C[1:n-7])} \quad (9.21)$$



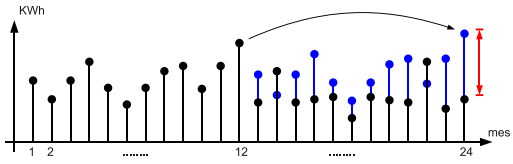
3. Cociente entre el valor medio en los últimos 12 meses y el pasado

$$car3 = \frac{mean(C[n-12:n])}{mean(C[1:n-13])} \quad (9.22)$$



4. Norma de la diferencia entre el consumo esperado y el consumo real

$$car4 = \sqrt{\sum_{i=n-11}^{i=n} (C(i) - \alpha C(i-12))^2} \quad (9.23)$$



5. Diferencia en los espectros

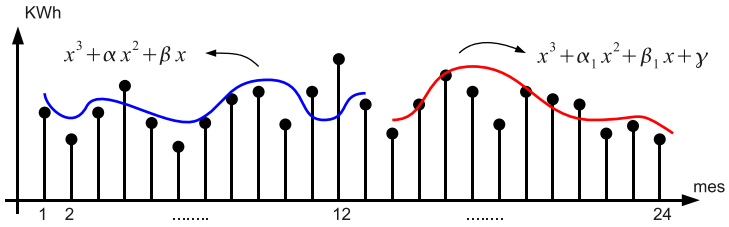
$$car5 = \|FFT\{C_{actual}\}(1) - FFT\{C_{medio}\}(1)\| \quad (9.24)$$
$$car6 = \|FFT\{C_{actual}\}(2) - FFT\{C_{medio}\}(2)\| \quad (9.25)$$
$$car7 = \|FFT\{C_{actual}\}(3) - FFT\{C_{medio}\}(3)\| \quad (9.26)$$

6. Diferencia en los coeficientes wavelet

$$car8 = |cA2_u(1)| - |cA2_p(1)| \quad (9.27)$$
$$car9 = |cA2_u(2)| - |cA2_p(2)| \quad (9.28)$$
$$car10 = |cA2_u(3)| - |cA2_p(3)| \quad (9.29)$$
$$car11 = |cA2_u(4)| - |cA2_p(4)| \quad (9.30)$$
$$car12 = |cA2_u(5)| - |cA2_p(5)| \quad (9.31)$$
$$car13 = \|cD2_u - cD2_p\| \quad (9.32)$$
$$car14 = \|cD1_u - cD1_p\| \quad (9.33)$$

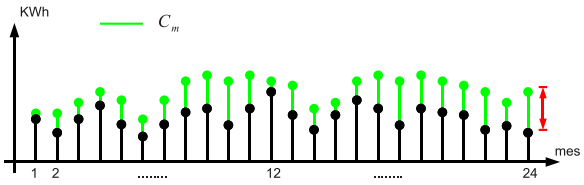
7. Diferencia en el ajuste de un polinomio de grado n

$$car\{15,16,17,18,19\} = polyfit(C_{ano\ n}) - \frac{1}{n-1}\sum_{i=1}^{n-1} polyfit(C_{ano\ i}) \quad (9.34)$$



8. Distancia al consumo medio

$$car20 = \left\|\vec{C} - \vec{C}_m\right\| \quad (9.35)$$



9. Comparación de la varianza del consumo

$$car21 = \frac{var\,(C_N)}{var\left(\frac{1}{N-1}\sum_{i=1}^{N-1} C_i\right)} \quad (9.36)$$

10. Varianza del consumo

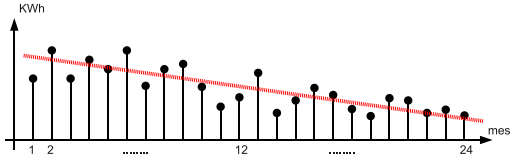
$$car22 = \frac{var\,(C_N)}{var\,(C_m)} \quad (9.37)$$

11. Coeficientes de Fourier

$$car\{23,24,25,26,27\} = \left\|FFT(C)_{\{1,2,3,4,5\}}\right\| \quad (9.38)$$

12. Pendiente de la recta que ajusta al consumo

$$car28 = polyfit(C,1) \quad (9.39)$$



CAPÍTULO 10

Métodos de selección implementados

En cualquier problema de reconocimiento de patrones es muy importante la caracterización del problema. El éxito del algoritmo final depende en buena medida de cómo se representan los patrones, es decir, cuales son las características que se extraen de los elementos, en nuestro caso, de los consumos eléctricos a analizar. Como se detalló en el capítulo 5 no solo es importante la calidad de las características (qué tan bien describan a las clases) sino que hay que tener muy en cuenta la cantidad de las mismas, y al contrario de lo que parecería intuitivo, a veces más no es necesariamente mejor. Características redundantes o irrelevantes puede que no aporten en la clasificación y que incluso perjudiquen la performance final. Esta es la principal causa por la que se decidió agregar una instancia de selección de características a nuestro problema.

Es importante recordar que al elegir un método de selección de características, dos elecciones están implícitas, el método de búsqueda y el método de evaluación. Nuestro objetivo va a consistir en, a partir de un grupo de *d* características, determinar cual es el *mejor* subconjunto de tamaño *p*. El método de búsqueda se refiere a cómo realizamos esa búsqueda, es decir, como recorremos todos los posibles subconjuntos para evaluarlos. Por otro lado, el método de evaluación es cómo evaluamos cada subconjunto para determinar cuál es el *mejor*.

Para elegir los métodos de selección a utilizar se estudiaron las diferentes posibilidades, evaluando su validez en el contexto de este proyecto. Para esto, se tiene en cuenta la naturaleza del problema, las características disponibles y los clasificadores considerados. Se realizo este proceso mediante el uso del programa Weka[a] en su versión 3.7.2. Weka es un proyecto de la universidad de Waikato, Nueva Zelanda, en la cual se desarrolló un software en java que agrupa una gran cantidad de métodos relacionados con reconocimiento de patrones entre los cuales se encuentran métodos de selección de características. Una vez que se cuenta con los subconjuntos de características que son salida de los métodos seleccionados, se probaron cada una de ellas con los 4 clasificadores

[a]Por más información de Weka dirigirse a http://www.cs.waikato.ac.nz/~ml/weka

que se implementaron y se muestran en el capítulo 11. Estos son: CS-SVM, One-Class SVM, OPF y el árbol C4.5

10.1. Métodos de evaluación utilizados

Como se vió en la sección 5.3 existen 2 grandes grupos de métodos, los *wrapper* y los *filter*. Los primeros son los que utilizan, para determinar qué tan bueno es un subconjunto, la performance de un clasificador utilizando dicho subconjunto. Hay que especificar qué clasificador usar y sus parámetros en caso de que sea necesario. Por otro lado los segundos solo utilizan la estructura de los datos para determinar cuáles son las características que mejor separan las cases.

De los métodos *filter* se utilizo el **CfsSubsetEval**. Este algoritmo evalúa la habilidad que tiene cada característica de predecir las clases tomando en cuenta el grado de redundancia entre ellas. Prefiere conjuntos altamente correlacionados con las clases y con baja correlación entre las características.

De la clase *wrapper* se utilizo la selección con los clasificadores C-SVM y el C4.5 (árbol de decisión), pues éstos se utilizarán en la clasificación. A su vez se utilizó *wrapper* con vecino más cercano, por su similitud con OPF, no estando éste último implementado en weka. Es importante tener en cuenta que cuando utilizamos un método *wrapper* se debe explicitar a su vez cual es el valor que se busca maximizar, o dicho de otra manera cual es la función de mérito que definimos. En nuestro caso utilizamos dos variantes, justificadas en lo que va del texto, el *Accuracy* y el *F-value*. La primera, al ser la medida estandar de performance en estos problemas es casi una obligación usarla. Sin embargo, al estar frente a un problema de clases desbalanceadas con las particularidades que esto implica (descriptas en el capítulo 4), una de las estrategias más convenientes es maximizar el *F-value*.

Los métodos de evaluación utilizados en este proyecto fueron:

- *Wrapper* con el clasificador **C-SVM** buscando maximizar tanto el *accuracy* como el *F-value*.
- *Wrapper* con el clasificador **C4.5** buscando maximizar tanto el *accuracy* como el *F-value*.
- *Wrapper* sensible al costo con el clasificador **C4.5** buscando maximizar el *accuracy*. Se aplico esta variación al método para analizar la incidencia que podia tener utilizar el árbol asignándole pesos distintos a las clases y así palear el problema de clases desbalanceadas.
- *Wrapper* con el clasificador **Vecino mas cercano**.
- CfsSubsetEval como método *filter*, ya que vimos que se pueden obtener buenos resultados.

10.2. Métodos de búsqueda implementados

El único método que garantiza encontrar el subconjunto óptimo para cualquier problema es el de búsqueda exhaustiva. Sin embargo el costo computacional que representa evaluar todas las posibilidades es demasiado alto. Aún teniendo en cuenta que la selección es algo que se va a realizar solo una vez para conjuntos muy grandes o algoritmos de evaluación muy pesados se vuelve prohibitivo. La única alternativa para llegar a un conjunto óptimo que no sea la antes mencionado es el método *branch and bound*, pero éste necesita que el mérito de los subconjuntos sea proporcional con la cantidad de características, algo que no podemos saber a priori en este problema.

De los métodos subóptimos que tratamos en el capítulo 5 el que vamos a utilizar para la mayoría de las corridas es el *bestfirst*. Este es un método un poco más exhaustivo que las búsquedas secuenciales y por lo tanto más costoso, pero teniendo en cuenta que nos llevará a una mejor solución, y que se correrá solo una vez con cada método de evaluación, asumiremos los costos computacionales, que aunque sean grandes no son prohibitivos.

La única excepción será con el método de evaluación CfsSubsetEval, el cual al ser muy rápido, nos tomaremos la libertad de realizar una búsqueda exhaustiva. Esto sería inviable en los métodos *wrapper* ya que habría que entrenar y testear un clasificador para cada combinación posible de características.

10.3. Resultados

En esta sección se presentarán primero los métodos de selección de características que se probaron, debido a lo planteado en las secciones anteriores. Para cada uno de ellos se mostrará el conjunto de características seleccionado, así como la notación que usaremos para diferenciar los métodos. Para finalizar se presentan los resultados obtenidos con cada uno de los clasificadores usados sobre los conjuntos de características seleccionados.

Notación

Para presentar de manera compacta los resultados obtenidos, se trabajará con abreviaciones que se detallan a continuación.

- *A* corresponde a las características Car-A definidas en la sección 8.3 (pag. 83) y hacen referencia al conjunto usado en el trabajo previo de Kosut y Alcetegaray [10]
- *C* corresponde a las características Car-C definidas en la sección 8.3 (pag. 83) las cuales corresponden a los consumos en sí mismos.
- *DeCA* corresponde a las características Car-D definidas en la sección 8.3 (pag. 83, las cuales se refieren a las desarrolladas en este proyecto)

- *DeCA Cfssubseteval*
  Se selecciona un subconjunto de las características DeCA definidas en el capitulo 9 (pag. 85) utilizando el método Cfssubseteval implementado en weka. Se utiliza como estrategias de búsqueda bestfirst y búsqueda exhaustiva; ambos arrojan el mismo subconjunto [b].
  Las características seleccionadas son $1,\ 2,\ 3,\ 12,\ 20,\ 28$.

- *DeCA wrapper-A C-SVM BF*
  Se toma un subconjunto de las características de DeCA utilizando el método wrapper con el clasificador C-SVM implementado en weka. Se busca maximizar el Accuracy y se utiliza como método de búsqueda bestfirst.
  Las características seleccionadas son $1,\ 2,\ 3,\ 4,\ 11,\ 23,\ 28$.

- *DeCA wrapper-Fv C-SVM BF*
  Se toma un subconjunto de las características DeCA utilizando el método wrapper con el clasificador C-SVM implementado en weka. Se busca maximizar F-value y se utiliza como método de búsqueda bestfirst.
  Las características seleccionadas son $1,\ 3,\ 20,\ 23,\ 28$. Estas características son seleccionadas como las mejores en las 3 validaciones cruzadas que se realizan, específicamente las seleccionadas en al menos 2 de las 3 validaciones.

- *DeCA wrapper-FvA C-SVM BF*
  Consiste en el mismo procedimiento que en el set anterior pero considerando un criterio más amplio. Se incluyen todas las características que son seleccionadas en al menos una de las corridas de la validación.
  Las características seleccionadas son: $\{1,\ 3,\ 20,\ 23,\ 28\} + \{2,\ 11,\ 21,\ 24\}$.

- *DeCA wrapper-CS-A tree*
  Se toma un subconjunto de las características DeCA utilizando el método wrapper cost sensitive implementado en weka , utilizando como clasificador el árbol J48 (C4.5) y como método de búsqueda bestfirst.
  Las características seleccionadas son $1,\ 28,\ 23,\ 2,\ 4,\ 17,\ 24$.

- *DeCA wrapper-A tree*
  Se toma un subconjunto de las características DeCA utilizando el método wrapper implementado en weka. Se busca maximizar la Accuracy utilizando como clasificador el árbol J48 (C4.5) y como método de búsqueda bestfirst.
  Las características seleccionadas son $2,\ 7,\ 11,\ 16,\ 28$.

- *DeCA wrapper-Fv tree*
  Se toma un subconjunto de las características DeCA utilizando el método wrapper implementado en weka. Se busca maximizar F-value utilizando como clasificador el árbol J48 (C4.5) como método de búsqueda bestfirst.
  Las características seleccionadas son $3,\ 4,\ 5,\ 11,\ 23,\ 24,\ 28$.

- *DeCA wrapper 1-NN*
  Se selecciona un subconjunto de la base DeCA utilizando el método wrapper con el

[b] Esto es algo muy positivo ya que indica que, por lo menos en este caso, el método bestfirst llega al subjonjunto óptimo al coincidir con la búsqueda exhaustiva

clasificador 1-NN (vecino más cercano), buscando maximizar el F-value. El método de búsqueda es el bestfirst.
Las características seleccionadas son $1, 3, 12, 20, 22, 23, 28$.

Performance

En la siguiente tabla se muestran los resultado de la performance de los distintos clasificadores usando los conjuntos de característica seleccionados. Se muestran el *accuracy* y el *F-value*, el primero por tratarse de una medida estándar para evaluar el desempeño de un clasificador y el segundo por tratarse del valor que, en nuestro caso, mejor describe la potencialidad de la herramienta.

Clasificador / Características	One-SVM		C-SVM		CS-SVM		OPF		C4.5	
	Acc	Fval	Acc	Fval	Acc	Fval	Acc	Fval	Acc	Fval
A	84,7	45,8	86,8	43,1	86,7	45,6	78,9	41	87,4	54
C	80,9	47,5	86,1	32,1	88,1	50,2	80	47,9	83,7	42
DeCA	81,9	45,0	87,71	49,42	88,2	49,6	72,5	42,4	87	53
DeCA Cfssubseteval	83,6	52,8	85,9	42,7	85,6	52,6	77	39,6	87,4	51,9
DeCA wrapper-A C-SVM BF	82,3	46,1	85,4	41,4	84,6	55,9	75	40,8	86,8	51
DeCA wrapper-Fv C-SVM BF	76,1	44,5	85,8	41,3	86,6	57,1	81,6	40,6	87,1	51
DeCA wrapper-FvA C-SVM BF	83,1	46,7	86,9	45,7	85,5	56,3	75,6	44,2	86,5	50
DeCA wrapper-CS-A tree	82,5	44,5	87,0	45,1	85,6	54,7	78	39,8	86,5	50
DeCA wrapper-A tree	81,7	47,2	86,5	33,9	87,4	45,1	77	38,4	87,3	52
DeCA wrapper-Fv tree	83,2	45,9	88,4	50,4	88,5	51,7	78,9	42,1	87,8	52,7
DeCA wrapper 1-NN	70,6	41,9	88,0	50,0	87,5	48,3	79	42,4	87	51,5

Analizando esta tabla podemos decidir cuales son las mejores características para cada clasificador

1. Para el clasificador **One Class SVM** el subconjunto de características que maximizó el F-value fue el **Deca Cfssubseteval**. Estas son las características obtenidas usando el método de evaluación Cfssubseteval (1,2,3,12,20,28)

2. Para el clasificador **CS-SVM** el mejor set de características en cuanto al F-value fue el **Deca wrapper Fv C-SVM BF**, es decir, el método wrapper usando C-SVM y maximizando el F-value. El método de búsqueda es el bestfirst. A pesar de éste resultado, y teniendo en cuenta que la diferencia de performance con el conjunto ampliado no era tanta, decidimos usar este último (1,2,3,11,20,21,23,24,28).

3. Si miramos el clasificador **OPF** vemos que la mejor performance (siempre mirando el F-value) se da con el set de características **C**, o sea usando como las características los consumos.

4. Por último, para el árbol **C4.5** las características con mejor performance respecto al F-value son las **A**. Estas son las características usadas en el proyecto de la materia de Int al reconocimiento de patrones [10]

Como se puede ver, cada clasificador tiene un conjunto de características óptimo distinto, lo que nos plantea la siguiente pregunta: ¿Debemos tomar características distintas para cada clasificador o tomar la que da mejor performance en promedio? La respuesta a esta pregunta sale del último paso del algoritmo, la combinación de clasificadores. Es sabido que la combinación de clasificadores es mejor si éstos son lo más independiente posible[20][23] . La diversidad de los clasificadores es algo que influye positivamente en la combinación[43]. Por esta razón es que vamos a tomar para cada clasificador las características con que se obtienen los mejores resultados.

CAPÍTULO 11

Clasificadores

11.1. Consideraciones Generales

En el capítulo 6 se introdujeron los fundamentos teóricos de 4 clasificadores, SVM One-Class, CS-SVM, OPF y C4.5. Tanto One Class SVM como CS-SVM están basados en la teoría de SVM (Support Vector Machine), su objetivo final es hallar ciertos parámetros que determinen un modelo óptimo, por lo que las técnicas llevadas a cabo para desarrollarlo y determinar estos modelos óptimos tienen varios aspectos en común. Por esta razón es pertinente comenzar con una introducción sobre mecanismos utilizados en el desarrollo de ambos clasificadores.

Antes de comenzar con la introducción, recordamos las ecuaciones que describen a ambos problemas:

- **One Class SVM**:

$$\min_{\omega \in \mathcal{H},\, \zeta_i \in \mathbb{R},\, \rho \in \mathbb{R}} \frac{1}{2} \|\omega\|^2 + \frac{1}{\nu\, l} \sum_{i=1}^{i=l} \zeta_i - \rho \tag{11.1}$$

$$\text{sujeto a las restricciones } \langle \omega, \Phi(x_i) \rangle \geq \rho - \zeta_i, \quad \zeta_i \geq 0 \tag{11.2}$$

  Donde $\nu \in (0, 1]$ es un parámetro que regula el compromiso entre cometer errores y obtener mayor separación de la frontera con el origen.

- **CS-SVM**:

$$\Phi(\omega, \zeta_i) = \text{mín} \left\{ \frac{1}{2} \|\omega\|^2 + \sum_{i/y_i=1} C^+ \zeta_i + \sum_{i/y_i=-1} C^- \zeta_i \right\} \tag{11.3}$$

$$\text{sujeto a las restricciones } y_i(\langle \omega, x \rangle + b) \geq 1 - \zeta_i, \;\; i = 1, 2, ..., n \tag{11.4}$$

  Donde ζ_i para $i = 1, 2, ..., n$ son variables no negativas.

11.1.1. Elección del Kernel

Para mapear el vector de entrada x a un espacio de características de mayor dimensión $\mathcal{H}$ dotado de producto interno para ambos clasificadores se decidió utilizar una función de kernel basada en la distancia Euclideana, la función de kernel en base radial (RBF)

$$K(x_i, x_j) = e^{-\gamma \|x_i - x_j\|^2} \tag{11.5}$$

donde el parámetro γ controla el ancho de la función. El kernel RBF induce un espacio de kernel de dimensión infinita en donde todos los vectores imágenes tienen la misma norma. Se tomó esta decisión pues es uno de los kernels más utilizados en la práctica y además ya fue utilizado en otros trabajos [10][30] de detección de fraude en energía eléctrica con buenos resultados.

11.1.2. Método de Validación Cruzada

El paso siguiente luego de determinar el tipo de kernel que se va a utilizar, consiste en determinar los parámetros óptimos que definirán el modelo del clasificador. Estos parámetros dependen naturalmente del kernel que se utilice (γ para RBF) y del tipo de SVM que se está implementando, en One Class SVM se debe determinar el parámetro de penalización de error ν mientras que en CS-SVM, C^+ y C^-.

Determinar el valor de estos parámetros de una manera adecuada es crucial a la hora de obtener buenos modelos para los clasificadores, en la figura 11.1 se ilustra cómo varía la frontera de decisión al variar γ. En la figura 11.2 se puede observar cómo afecta la variación de ν a la hora de determinar el margen y el número de vectores soporte. De qué manera es alterado el desempeño de nuestro clasificador al variar simultáneamente el parámetro de penalización de error y del kernel no es sencillo de predecir y será abordado en lo que sigue.

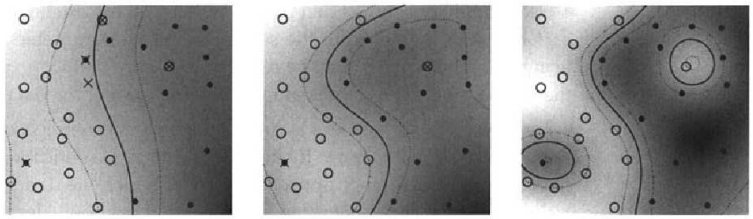


Figura 11.1: Figura a los efectos ilustrativos de como varía la frontera de decisión para un kernel Gaussiano al variar γ. De izquierda a derecha se incrementa γ. Se puede observar que para valores pequeños de γ la frontera es más lineal y los datos no pueden ser separados sin error, en el otro extremo para valores elevados obtenemos una frontera más ajustada a los datos donde se obtiene un comportamiento no lineal y menos errores en la clasificación. Si reducimos mucho el ancho del kernel debemos tener precauciones de no obtener una curva muy sobre adaptada a los datos de entrenamiento. Imagen tomada de [38]

El primer paso para determinar los parámetros consiste en dividir la base de datos en dos, una de entrenamiento y otra de test. La base de entrenamiento es utilizada para

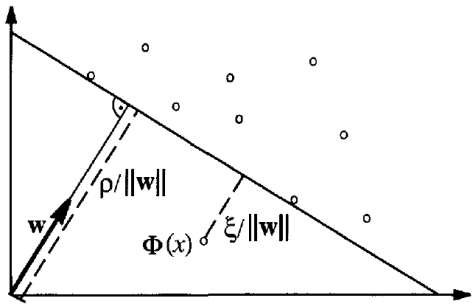


Figura 11.2: Ejemplo en el caso en que trabajamos con muestras en $\mathbb{R}^2$ para ilustrar el compromiso entre obtener un margen amplio y obtener pocos errores. Imagen tomada de [38, pag:232]

probar distintos valores del parámetro de penalización de error y γ para así obtener el modelo óptimo del clasificador. Se realizó en la base de entrenamiento una validación cruzada partiendo el conjunto en 5 partes como se explica en sección 6.2.3 para obtener los mejores parámetros

11.1.3. Criterio de diseño y Medidas de Performance

En la búsqueda de los parámetros óptimos se debe definir un criterio de diseño para medir el error de clasificación, es decir, traducir de alguna forma la importancia que se le da a la cantidad de consumos anómalos y normales mal clasificados. Estos criterios tienen el objetivo de lograr que la clasificación sea sensible al costo de forma de minimizar el riesgo de mala clasificación.

En el capítulo 4 se hizo una introducción a los problemas con clases desbalanceadas, las dificultades que presenta tratar este problema con técnicas de clasificación más comunes así como también nuevas formas (o enfoques) de enfrentarlos. Allí se introdujo la posibilidad de utilizar costos en la toma de decisiones a partir de la matriz de costos, como se puede observar en la tabla 11.1, donde el costo de clasificar una muestra de la clase j como de la clase i corresponde a la entrada λ_{ij} de la matriz. Debemos tener en cuenta que el objetivo en los problemas de clasificación sensibles al costo es minimizar el costo en los errores de clasificación.

	Class i	Class j
Class i	0	λ_{ij}
Class j	λ_{ji}	0

Cuadro 11.1: Matriz de Costo

En la sección 4.3 (pag. 38) se introdujeron diferentes medidas de performance de clasificadores para casos en donde existe un determinado desbalance de clases. Una herramienta sumamente útil que resume los resultados de clasificación es la matriz de confusion 11.2. Allí se puede ver como el clasificador o clasificadores utilizados etiquetan

las muestras, por ejemplo, TP denota la cantidad de muestras positivas mal clasificadas. Antes de continuar debemos introducir la nomenclatura utilizada:

- Positive refiere a las muestras de la clase **anómalos** (clase minoritaria).
- Negative refiere a las muestras de la clase **normal** (clase mayoritaria).

La gran mayoría de las medidas de performance para problemas de dos clases parten de la matriz de confusión que se observa en la tabla 11.2. A partir de la misma se obtienen 4 simples medidas, TP (True Positive) y TN (True Negative) denotan el numero de muestras positivas y negativas correctamente clasificadas, mientras que FP (False Positive) y FN (False Negative) refieren al numero de muestras positivas y negativas incorrectamente clasificadas. Las medidas de performance más utilizadas en los sistemas de aprendizaje son

$$err = \frac{FP + FN}{TP + FN + TN + FP} \qquad acc = \frac{TP + TN}{TP + FN + TN + FP}$$

.

Sin embargo, se ha demostrado que cuando existe un gran desbalance en la distribución de clases, estas medidas no son apropiadas ya que no toman en consideración los costos de clasificar incorrectamente las distintas clases, son extremadamente sesgadas para favorecer a la clase mayoritaria y son sensibles a los sesgos de clase. Por ejemplo, si tenemos un caso donde el 1 % de las muestras pertenecen a la clase minoritaria, al clasificar todas las muestras como de la clase mayoritaria se obtiene un ”accuracy” del 99 % pero sin lograr clasificar correctamente ninguna muestra de la clase minoritaria. A partir de estas medidas y de la matriz de confusión se puede definir criterios como *precisión* (referido a la precisión en alguna de las dos clases), *recall*, *ROC*, *AUC* (área debajo de la curva ROC), F_{value}, *MGM* (máximo promedio geométrico de la precision en la clase de la mayoría y de la minoría), *MS* (suma máxima de precisión).

	Positive	Negative
Positive	TP (True Positive)	FN (False Negative)
Negative	FP (False Positive)	TN (True Negative)

Cuadro 11.2: Matriz de Confusión

En esta sección solo se definen las medidas que se utilizarán tanto para la etapa de diseño como para medir la performance de los distintos clasificadores. Estas son:

- $Recall = \frac{TP}{TP + FN}$
- $Precision = \frac{TP}{TP + FP}$
- $F_{value} = \frac{(1 + \beta^2) Recall \times Precision}{\beta^2\, Recall + Precision}$.
- $MS = Accuracy_+ Accuracy_-$

Se utilizará el F_{value} como medida principal para evaluar el desempeño, éste combina Precisión y Recall, y permite controlar el peso de cada uno por separado variando el valor de β. Este indicador es uno de los más utilizados en el abordaje de problemas con clases debalanceadas [1][3][17]. Dependiendo de los problemas particulares que se abordan, se puede dar distinta relevancia a $Recall$ frente a $Precision$ o vice versa. A modo de ejemplo [6], en un esquema de clasificación en el que se pretende separar páginas de interés o relevantes, como es el caso de un buscador web, el usuario desea que todas los sitios sugeridos en la primer página de resultados sean relevantes. Esto se logra buscando alta $Precision$, en ese caso no es tan importante que el $Recall$ sea bajo pues de todos modos el usuario no puede abarcar todas las páginas relevantes. Sí es importante que aquellas que va a observar sean relevantes. En contraste otras aplicaciones requieren altos niveles de $Recall$ (sin importar el costo sobre la $Precision$), un ejemplo de éstas son las búsquedas de archivos en un disco duro, donde se desea encontrar a todos los archivos que determinados patrones [6].

En conclusión, el F_{value} es una medida que se adapta al problema de clases debalanceadas, y además, es lo suficientemente versátil como para poder abarcar (fijando adecuadamente β) distintas clases de problemas. Por estas razones el uso del F_{value} como medida de performance parece más que justificado. Sin embargo, no queda claro aún el valor de β que mejor se adapta al problema que se pretende abordar. Fijaremos como primer abordaje al problema $\beta = 1$, esto implica que daremos igual importancia al $Recall$ que a $Precision$. Se puede cambiar el valor de β en futuros abordajes al problema, sin hacer cambios importantes al software implementado, simplemente redefiniendo el parámetro “beta”. El ajuste final del valor de β óptimo debe hacerse en campo. Es decir, al cambiar el valor del mismo modificamos el punto de trabajo, y esto debe hacerse en función de los recursos e intereses particulares que se tenga a la hora de utilizar la herramienta que aquí se desarrolla. A modo de ejemplo, si se posee gran capacidad para realizar inspecciones podemos ponderar el $Recall$, esto generará un índice de detección de consumos anómalos más elevado, pero, aumentará también el número de FP (disminuyendo la $Precision$). En contraste, si poseemos pocas unidades para realizar inspecciones, podríamos optar como estrategia detectar pocos consumos anómalos (pagando el costo de más índice de FN) pero teniendo una alta $Precision$, es decir detectamos un conjunto menor, pero ese conjunto no tiene casi FP.

11.1.4. Base de Datos

Para analizar el comportamiento de los distintos clasificadores, estudiar su validez frente al problema de detección de consumos anómalos a partir de las medidas de performance antes mencionados, y determinar el modelo óptimo para cada uno de ellos, es necesario utilizar una base de datos que represente las distintas clases del problema (anómalos y normales). Se decidió utilizar la **Base 2** la cual esta compuesta por 1504 suministros industriales de distinta índole (almacén, supermercado, autoservicio y almacén/vivienda) ya que la misma está formada por un numero de suministros suficientes para caracterizar a ambas clases (aunque existe un gran desbalance de clases [a]) y porque fueron elegidos de forma aleatoria por los técnicos de UTE.

[a] Para la Base 2 el desbalance de clases es de aproximadamente 1:10, esto es, hay 10 normales por cada anómalo

11.2. One Class SVM

Para determinar ν y γ se puede utilizar el siguiente procedimiento (sugerido en [38]):

1. Dividimos la base de datos en p partes iguales, llamemos a cada base como B_i con $i = \{1, 2, ..., p\}$
2. Tomamos $B_{te} = B_1$ como base de test y $B_{tr} = B_2 \cup B_3 \cup ... \cup B_p$ como base de entrenamiento.
3. Para cada $\nu \in U = \{\nu_1, \nu_2.....\nu_m\}$ y $\gamma \in S = \{\sigma_1, \sigma_2.....\sigma_l\}$ entrenamos el clasificador con la base B_{tr} y obtenemos una frontera de decisión f.
4. Con f clasificamos los vectores de la base B_{te} y comparamos la etiqueta obtenida con la etiqueta de cada muestra. De la comparación anterior obtenemos el valor de F_{value} estimado para ese combinación de ν y γ, llamado $F_{value_1}(\nu, \gamma)$.
5. Repetimos el procedimiento anterior considerando $B_{te} = B_2$ y la unión de las bases restantes como B_{tr} obteniendo $F_{value_2}(\nu, \gamma)$, luego $B_{te} = B_3$ y así sucesivamente hasta haber completado las p iteraciones.
6. Contamos entonces con una estimación para el F_{value} del clasificador para cada $\nu \in U$ y $\gamma \in S$ en cada uno de los p casos $(F_{value\,1},\ F_{value\,2},\ \ldots\ ,\ F_{value\,p}\ )$. Tomamos como F_{value} asociado a cada ν y γ el valor promedio de los F_{value} obtenidos en las validaciones cruzadas, $F_{value}(\nu, \gamma) = \frac{1}{p} \sum F_{value,ß}(\nu, \gamma)$.
7. Finalmente escogemos el valor de ν y γ con el que se obtiene el mayor F_{value} promedio: $(\nu_{final},\ \gamma_{final}) = \text{argmín}_{\nu \in U,\ \gamma \in S} \{F_{value}(\nu, \gamma)\}$

A priori no se conoce el rango en el que se encuentran los ν y σ que maximizan el F_{value} (ν esta acotado entre 0 y 1 por la forma en que está definido) por lo que se realiza una exploración del F_{value} al variar ν y σ en un rango bastante amplio recorrido exponencialmente. Al analizar el comportamiento de F_{value} para los valores elegidos de ν y γ 11.3 se observa que el rango acotado para ν y γ corresponde a la zona determinada por $\nu \in [10^{-7}, 1]$ y $\gamma \in [10^{-6}, 1]$ donde se obtienen los valores máximos de F_{value}.

El siguiente paso es evaluar la performance del clasificador y definir un modelo óptimo. Como se mencionó en la sección 11.1.4, la base de datos utilizada es la **Base 2**.

	Positive	Negative
Positive	86	78
Negative	79	823

Cuadro 11.3: Matriz de Confusión para One Class SVM

Medidas de Performance:

- $Accuracy$=85,27 %
- $Recall_+$=52,44 %

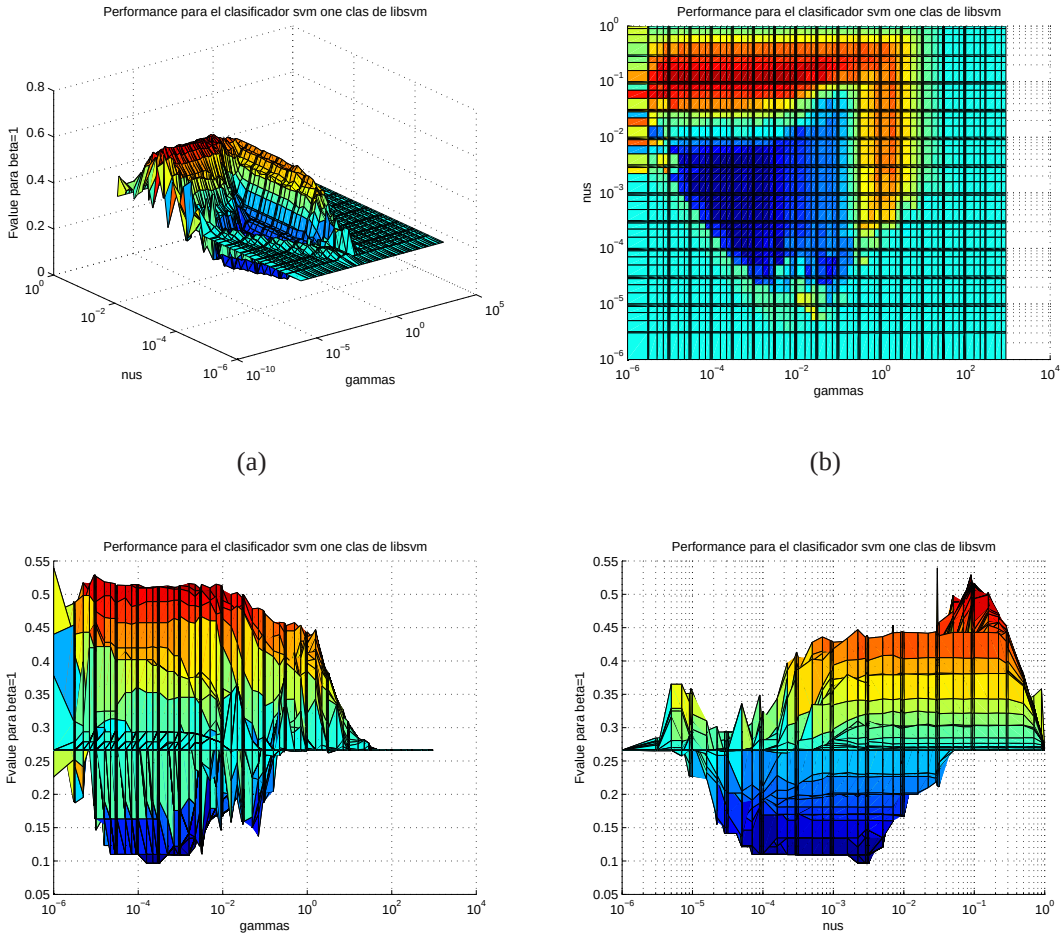


Figura 11.3: F_{value} promedio para $\nu \in [10^{-7}, 1]$ y $\gamma \in [10^{-6}, 10^{3}]$ para el conjunto de datos acotado. La escala es logarítmica en ν y γ

- $Precision_{+}$=52,12 %
- F_{value}=52,28 %

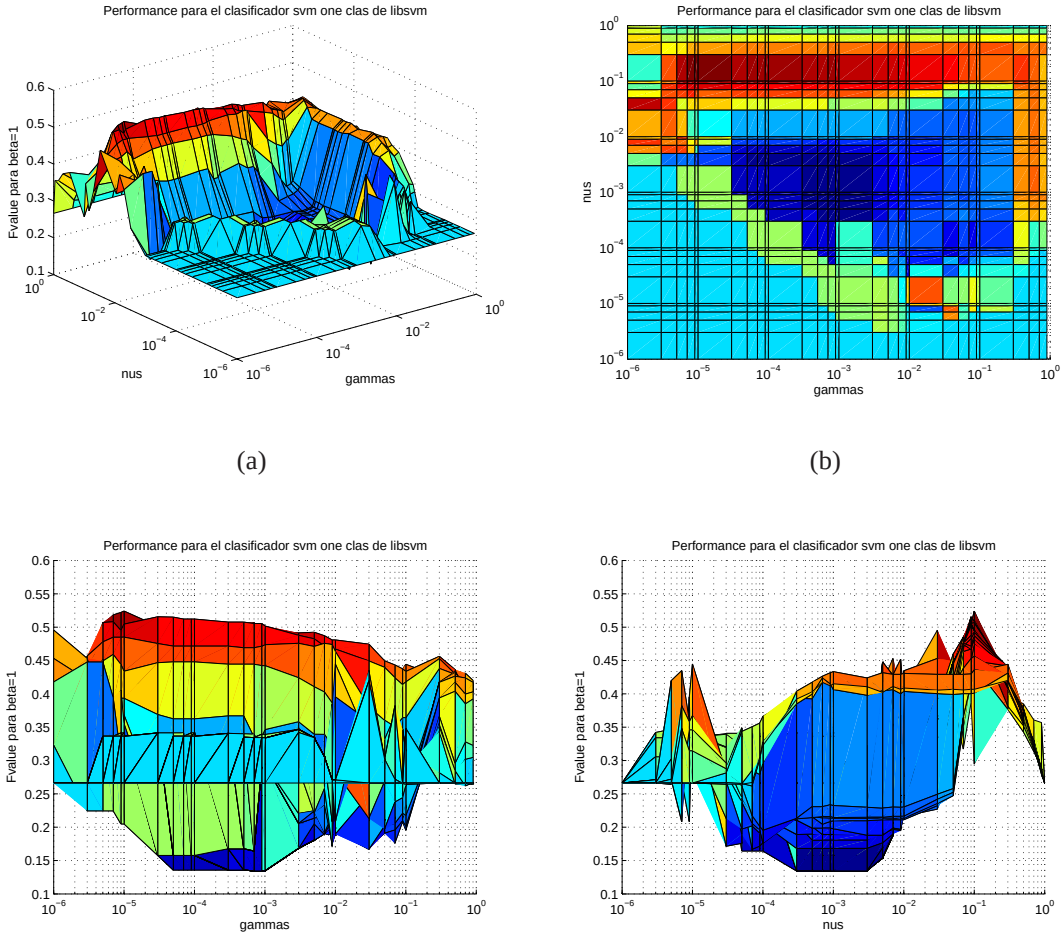


Figura 11.4: F_{value} promedio en función de ν y γ para el conjunto de datos acotado. La escala es logarítmica en C y γ

11.3. CS-SVM

Al igual que para One Class SVM, el procedimiento utilizado para determinar C^+_{opt}, C^-_{opt} y γ_{opt} esta basado en el sugerido en [38] utilizando validación cruzada y llevado a cabo de la siguiente manera:

1. Se determinan los conjuntos $C = [C_1, C_2, ..., C_n]$ y $\gamma = [\gamma_1, \gamma_2, ..., \gamma_m]$.

2. Se eligen $C_i \in C$ y $\gamma_j \in \gamma$ se divide la base de datos de entrenamiento en p partes iguales y se realizan p corridas de entrenamiento. Llamamos a cada base como B_i con $i = \{1, 2, ..., p\}$.

3. Se utiliza $B_{te} = B_1$ como base de test y $B_{tr} = B_2 \cup B_3 \cup ... \cup B_p$ como base de entrenamiento.

4. A partir de B_{tr}, C_i y γ_j se crea un modelo del clasificador. Como la relación entre las dos clases no es balanceada (hay una mayor cantidad de consumos normales que anómalos), al crear el modelo del clasificador CS-SVM se definen C^+ y C^- utilizando pesos de clase definidos como la relación entre el número de muestras de entrenamiento sobre el número de muestras de cada clase [30].

5. Con este modelo se clasifican los vectores de la base B_{te} y se compara la etiqueta obtenida con la etiqueta de cada muestra. De esta comparación se obtiene el F_{value} estimado para estos valores de C_i y γ_j que llamaremos $F_{value_1}(C_i, \gamma_j)$.

6. Se repite el procedimiento anterior considerando $B_{te} = B_2$ y la unión de las bases restantes como B_{tr} obteniendo $F_{value_2}(C_i, \gamma_j)$, luego $B_{te} = B_3$ y así sucesivamente hasta haber completado las p iteraciones.

7. Para el par de valores (C_i, γ_j) se cuenta con una estimación de los F_{value} del clasificador para cada validación cruzada. Se toma como F_{value} asociado a este par (C_i, γ_j), el valor promedio de los F_{value} obtenidos en las validaciones cruzadas, $F_{value}(C_i, \gamma_j) = \frac{1}{p} \sum F_{value_l}(C_i, \gamma_j)$.

8. Este mecanismo se repite combinando todos los valores de los conjuntos C y γ.

9. Finalmente, se elige como C_{opt} y γ_{opt} los valores con los cuales se obtiene el mayor F_{value} promedio. C^+_{opt} y C^-_{opt} se obtienen a partir de C_{opt} y los pesos de clase.

A priori no se conocen los valores óptimo C_{opt} y γ_{opt} por lo que se debe realizar una búsqueda exhaustiva procurando determinar un rango acotado cercano a la solución. Como primer paso se realiza una búsqueda de los parámetros C y γ que maximizan el F_{value} en un rango muy amplio recorrido exponencialmente, en particular se realiza esta búsqueda para $C \in [2^{-8}, 2^{15}]$ y $\gamma \in [2^{-15}, 2^{8}]$. Al analizar el comportamiento de F_{value} para los valores elegidos de C y γ (11.5) se observa que el rango acotado para C y γ corresponde a la zona definida por $C \in [10^{-4}, 1]$ y $\gamma \in [10^{-4}, 1]$ donde se obtienen los valores máximos de F_{value}.

Habiendo determinado la zona donde se encuentran los parámetros óptimos para definir el modelo del clasificador se deben realizar pruebas para evaluar la performance del mismo.

Medidas de Performance:

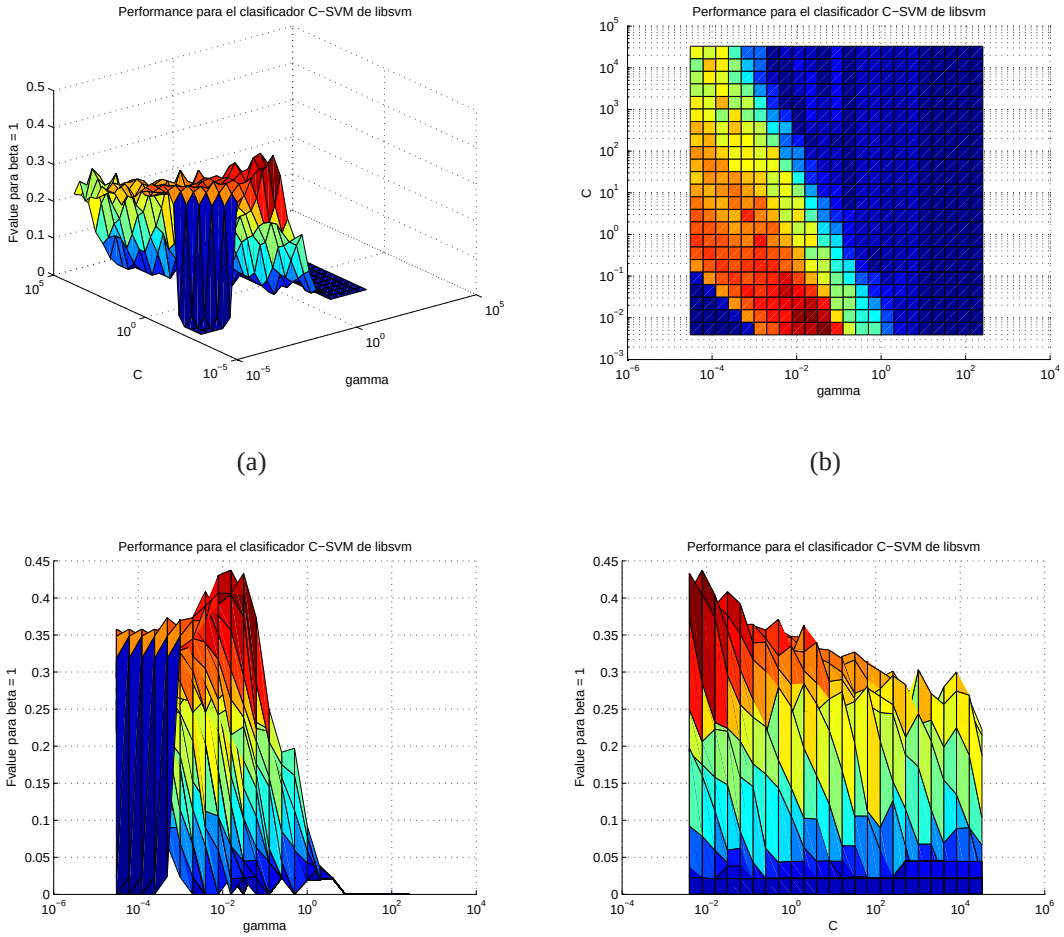
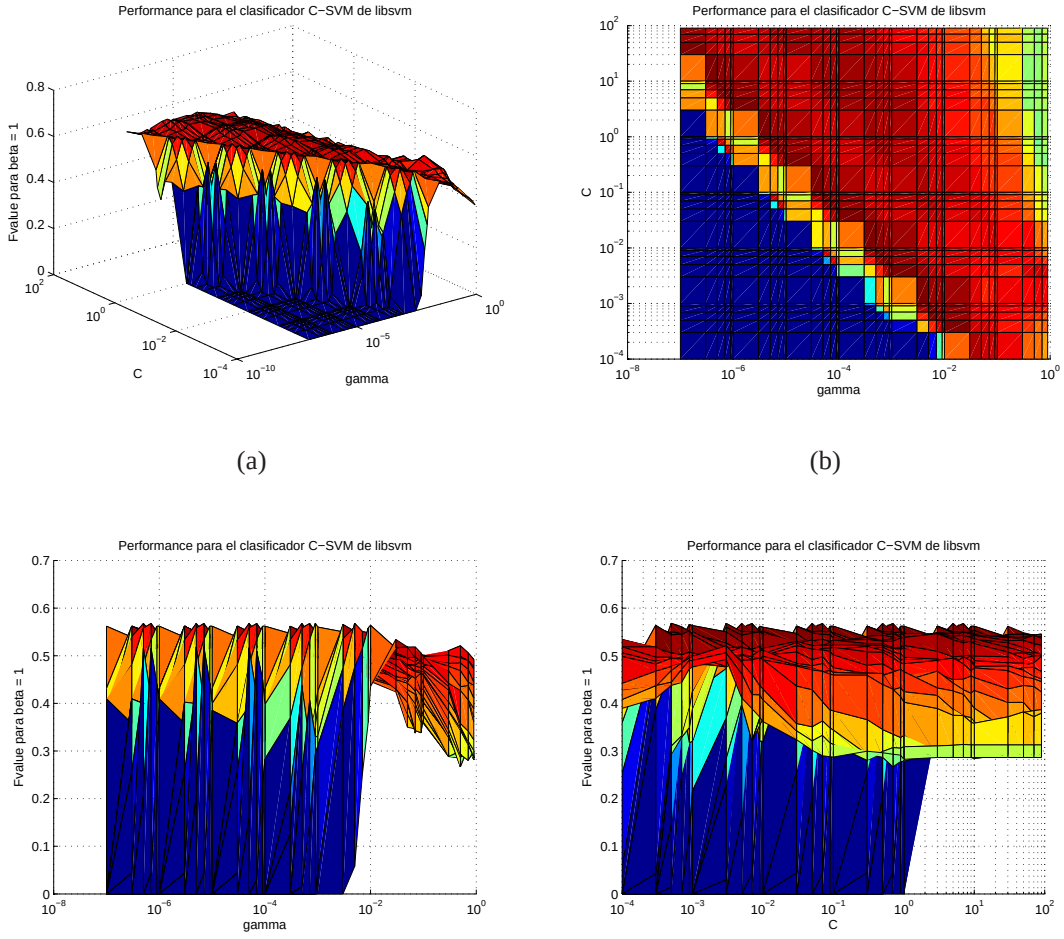


Figura 11.5: F_{value} promedio en función de C y γ para $C \in [2^{-8}, 2^{15}]$ y $\gamma \in [2^{-15}, 2^{8}]$. La escala es logarítmica en C y γ

	Positive	Negative
Positive	100	64
Negative	109	793

Cuadro 11.4: Matriz de Confusión para CS-SVM

- $Accuracy$=83,77 %
- $Recall_+$=60,98 %
- $Precision_+$=47,85 %
- F_{value}=53,62 %



(a) (b)

(c) (d)

Figura 11.6: F_{value} promedio en función de C y γ para el conjunto de datos acotado. La escala es logarítmica en C y γ

11.4. OPF

11.4.1. Consideraciones generales

Como se vió en la sección 6.3.1 la principal diferencia en la implementación de Optimum Path Forest (OPF) en relación a los clasificadores SVM es que OPF no tiene parámetros libres que deban ser determinados. Esto aporta una facilidad para la implementación del clasificador, pero sobre todo es una ventaja en cuando al tiempo necesario para el entrenamiento, no se precisa hacer ninguna búsqueda de parámetros óptimos como se vio en la sección 11.2 y la 11.3.

Por otro lado la desventaja de OPF es que no posee ninguna opción a priori para palear el desbalance de clases, como pueden ser costos diferentes para el error en cada clase (ver cuadro 11.1). Si recordamos el algoritmo de clasificación de OPF, vemos que cuando nos enfrentamos a un nuevo elemento para asignarle etiqueta lo asociamos al prototipo con costo de camino menor. Si tenemos en cuenta que el costo de camino es función de la distancia, parece lógico que si hay muchos más elementos de una clase que de otra, el clasificador etiquete a la mayoría de los patrones de prueba a la clase mayoritaria.

Como el objetivo principal es detectar a los elementos de la clase minoritaria tenemos que solucionar este hecho mediante alguna modificación. Con este fin, se encuentran los métodos que se describen en la sección 4.1, como puede ser submuestreo y sobremuestreo. El primero de estos tiene la desventaja de que al submuestrear se pierden datos, mientras el segundo tiene el problema de que se generan datos artificiales, con el riesgo que esto siempre significa. En este caso, como el desbalance no es excesivamente grande (de 10:1 aprox) y la base es de un tamaño considerable optamos por la idea de el submuestreo, ya que nos garantiza de todas formas una cantidad de datos no despreciable para trabajar.

Una vez que optamos por submuestrear, la siguiente pregunta es ¿En qué proporción submuestreamos? Aquí es donde aparece nuestro primer parámetro a la hora de diseñar el clasificador, al que llamaremos p_bal (proporción de balanceo) y definimos de la forma,

$$p\_bal = \frac{cantidad\ anomalos}{cantidad\ consumos}$$

Por ejemplo si p_bal=0,5 significa que hay la misma cantidad de anómalos que normales. Es importante hacer notar que el submuestreo se realiza únicamente en la etapa de entrenamiento, mientras que a la hora de evaluar el desempeño se mantiene la base completa.

11.4.2. Resultados

Para determinar como se comporta el clasificador con distintos valores de p_bal se ejecutaron pruebas con distintos valores del mismo. Los resultados que se muestran en

p_bal	Accuracy	Recall	Precision	Fvalue ($\beta = 1$)
0	84,4	37,2	49,2	42,4
0,2	84,4	43,9	49,3	46,4
0,3	81,6	53	42,2	47
0,4	81,4	61,6	42,8	50,5
0,5	77,9	65,85	37,5	47,8
0,6	73,7	69,5	33,1	44,8
0,7	70,3	80	31,5	45,2
0,8	61,3	83,5	26,2	40
0,9	52,3	88,4	22,87	36,3

Cuadro 11.5: Evaluación de OPF en función del balanceo.

la tabla 11.5 son para la base de datos 2 y las características seleccionadas en la etapa de selección de características.

Podemos ver claramente como a medida de que aumenta la proporción de anómalos en la etapa de entrenamiento (p_bal) el porcentaje de anómalos bien clasificados también aumenta, tal cual como lo esperamos. Sin embargo, algo que también se podía esperar, era que si colocábamos más anómalos en el entrenamiento iba a aumentar los FP, es decir los normales mal clasificados (normales clasificados como anómalos) disminuyendo así el indicador de precisión. También podemos asegurar, a partir de esta tabla, que si lo que queremos en nuestro problema es maximizar el F-value debemos tomar un valor de p_bal cercano al 0,4.

11.4.3. Software

Nuestro primer acercamiento a la técnica de OPF fue mediante el artículo de Caio César Oba Ramos et al. [9] el cual mostraba como con la técnica OPF era usada para la detección de fraudes en energía eléctrica. En este trabajo se nombraba el sitio donde está disponible el software OPF, dentro de la web del instituto de computación de la Universidad Estadual de Campiñas, Brasil. La página es `http://www.ic.unicamp.br/~afalcao/libopf/`. Desde allí se descarga el paquete LibOPF, el cual compilamos para UNIX y llamamos a los ejecutables desde matlab.

11.5. Árbol de Decisión

11.5.1. Parámetros del árbol

Como se vio en la sección 6.4, donde se abordo la teoría de los árboles de decisión, a la hora de implementar un clasificador de esta naturaleza, hay varios parámetros que determinar, los mismos se expondrán a continuación y se justificará la elección de cada uno

- *Particiones binarias*.
  La primera decisión es si a la hora de crecer el árbol permitimos que al hacer la división en un nodo salgan más de dos ramas o solo haremos particiones binarias.

Es fácil de mostrar, como lo hace Duda et al. en [12] que cualquier decisión múltiple se puede convertir en una concatenación de decisiones binarias, por lo tanto si decidimos que solo se puedan realizar decisiones binarias, no solo no perdemos generalidad sino que además estamos ganando simplicidad a la hora de la construcción. Por lo tanto, en el árbol implementado, cada nodo tendrá únicamente 2 sub-nodos hijos.

- *Criterio de crecimiento*.
  Otra decisión que se debe tomar en el crecimiento del árbol es qué criterio se utiliza para medir la impureza. Esto va a repercutir en cómo se elige la mejor característica para realizar la división en cada nodo. En la sección 6.4 explicamos las 3 más comunes, la impureza de Gini, la impureza de información (entropía) y la impureza de error de clasificación. Duda et al. en [12] demuestra que la impureza de error de clasificación puede, en algunos casos, ser ciega a algunas divisiones beneficiosas, por lo que generalmente no se utiliza a la hora del crecimiento [12]. En varias fuentes ([12][23] entre otros) aseguran que el árbol **C4.5** es el árbol más popular en la actualidad por lo tanto vamos a adoptar los criterios de éste. Esto implica fijar como criterio de crecimiento que vamos a utilizar es de *impureza de información*.

- *Criterio de parada y podado*
  A pesar de las variantes que se expresaron en el punto anterior, se cree que la variante de impureza que utilicemos no influye de sobremanera en el éxito del árbol [12], sino que lo que realmente importa es que método usemos para parar el crecimiento o como podamos el árbol una vez que ha crecido. También se comentó en la sección 6.4 que los criterios de parada producir el *efecto horizonte*, es decir, detener el crecimiento antes de un punto óptimo. Es por esto que vamos a optar por un criterio de podado a la hora de evitar el sobre-entrenamiento. Como optamos por utilizar el algoritmo de **C4.5**, vamos a elegir entonces el *podado pesimista* (ver sección 6.4).

Otro problema que hay que tener en cuenta es que al igual que el clasificador OPF, el árbol de decisión se ve influido por el desbalance de clases. Si en el conjunto de entrenamiento hay muchos más patrones de una clase, el algoritmo va a asignar, a la mayoría del espacio de características, la etiqueta de la clase mayoritaria. Por esto es que al igual que en OPF se submuestreará la clase mayoritaria para *balancear* las clases

11.5.2. Adaboost

En la sección 6.4, vimos que los árboles de decisión dependen mucho de los datos de entrenamiento, y esto lo hacía un algoritmo inestable: cambiar un poco los datos puede resultar en un árbol totalmente distinto. Esto es una propiedad que no es para nada deseada a la hora de construir un clasificador , sin embargo, esta diversidad de clasificadores, resulta tentador para realizar una combinación de varios de ellos. Es por esto que al árbol se le realiza un proceso de *adaboost*. El adaboost, que ya se explicó en la sección 7.2.2, consiste en generar muchas variantes del mismo clasificador variando la base de datos con la que se entrena. Una vez que se entrenan todos los clasificadores, se realiza una combinación de todos ellos y se toma como salida de ese método esta combinación. Se tomó 15 como el número de instancias para el algoritmo, como un buen equilibrio entre

precisión y costo computacional. También observando la figura 7.2 (mostrada en la pag. 73) se puede ver que a partir de 15 el error no decae de forma sensible.

11.5.3. Resultados

En esta sección se mostrará la performance del árbol variando el parámetro p_bal, el cual controla el grado de submuestreo.

p_bal	Accuracy	Recall	Precision	Fvalue ($\beta = 1$)
0	85,8	37,2	56	44,7
0,3	81,6	53,7	42,3	47,3
0,4	79,6	66,4	40,2	50,1
0,5	74,6	70,7	34,3	46,2

Cuadro 11.6: Evaluación del árbol en función del balanceo.

Al igual que en el caso de OPF si queremos mejorar la medida de F_{value} debemos seleccionar un valor de p_bal cercano a 0.4.

11.5.4. Software

Los algoritmos del árbol de decisión y del adaboost fueron extraídos del *PRTOOLS*, un toolbox de matlab gratuito diseñado especialmente para el área de reconocimiento de patrones. Por más información (o descargas del programa) consultar `http://prtools.org/`.

CAPÍTULO 12

Combinación

Antes de introducir los métodos de combinación de clasificadores implementados, se debe tener en cuenta la naturaleza de los clasificadores que se intenta combinar. Como se mencionó en el capitulo 7 las estrategias de combinación de clasificadores dependen en buena medida de la información que brinda la salida de cada uno de los clasificadores base. En este problema en particular, los clasificadores implementados son:

- One Class SVM
- WC-SVM (C-SVM con pesos distintos para las distintas clases).
- OPF (entrenado con y sin balanceo)
- Arbol C4.5 con AdaBoost.

Observemos brevemente las salidas disponibles en cada uno de estos clasificadores antes de plantear estrategias posibles de combinación. Para el árbol, se dispone para cada muestra de la base test, la clase a la que es asignada dicha muestra y la *probabilidad* de que la etiqueta sea correcta. Si bien el clasificador arroja una medida asociada a la confianza de la etiqueta que se esta asignando, esta medida (que en el algoritmo se llama probabilidad de pertenecer a la clase) se calcula en base a la cantidad de elementos en el nodo de una y otra clase de la base de entrenamiento. Por tal motivo desde el punto de vista formal la medida que se obtiene es (en el mejor de los casos) una **estimación** de la probabilidad de que la etiqueta sea correcta.

En el caso de CS-SVM, también obtenemos para cada muestra la probabilidad de pertenencia a una u otra clase. La etiqueta que se asigna a cada muestra corresponde a la clase con mayor probabilidad. Al igual que ocurre en el caso de árbol, se debe manejar con cierta precaución estas medidas, ya que son estimaciones que se realizan en función de la estructura de la base de datos utilizada para entrenar. En este caso la probabilidad de pertenecer a una u otra clase se calcula en función de la distancia a la frontera de decisión.

Para el clasificador OPF, la salida es únicamente la etiqueta a la que se asigna cada muestra de la base test, no se dispone de probabilidades de pertenencia a una u otra clase ni de indicadores que nos permitan tener idea de la confianza que se tiene sobre la etiqueta asignada.

Siguiendo con One Class SVM, nos encontramos en las mismas condiciones que para OPF, se cuenta con salida correspondiente a la etiqueta de la clase que se asigna para cada muestra de la base test pero ninguna otra información adicional. Vale la pena aclarar que en estos 2 últimos casos, el no poseer una medida diferenciada para cada muestra no depende tanto de la estructura del algorítmo de clasificación sino fundamentalmente de como fue implementado el software del mismo.

Además de la información que cada clasificador otorga para cada muestra, es importante a la hora de la combinación contar con una medida del desempeño global de cada uno de los clasificadores. Por esto es que se estima, utilizando validaciones cruzadas en la base de entrenamiento, los siguientes indicadores de performance:

$$Acc = \frac{TP + TN}{TP + FP + TN + FN} \tag{12.1}$$

$$Acc_{+} = \frac{TP}{TP + FN} \tag{12.2}$$

$$Acc_{-} = \frac{TN}{TN + FP} \tag{12.3}$$

$$Fvalue_{(\beta=1)} = \frac{Presicion \times Recall}{Presicion + Recall} \tag{12.4}$$

De este modo, si bien para algunos de los clasificadores no poseemos una medida individual de la confianza asociada a cada etiqueta, tenemos varios indicadores de la performance global.

12.1. Métodos propuestos

En la literatura y las publicaciones más recientes no se han encontrado estrategias de combinación de clasificadores que resuelvan de manera satisfactoria y con un fundamento formal el problema. Por otro lado no se encuentra consenso general en cuanto a las estrategias de combinación planteadas. En la mayoría de los casos las estrategias propuestas se basan en fundamentos intuitivos o formalizados para casos particulares y luego evaluados en problemas concretos [5][48][3]. Sin embargo es muy difícil encontrar respuesta a preguntas como: ¿Cómo se calculan los coeficientes para realizar un esquema de voto ponderado si se desea mejorar el F_{value} de la combinación? [a] ¿Cómo se combinan clasificadores donde se tiene medidas individuales de performance para algunos clasificadores y medidas globales para otros? ¿Qué estrategias de las utilizadas en casos normales se pueden utilizar con iguales resultados en problemas de clases desbalanceadas?

[a] No es igual a mejorar el Accurary como se demuestra en [23] para el caso de clasificadores independientes

A continuación presentamos algunos esquemas de combinación propuestos, basados en publicaciones que proponen estrategias para casos particulares y en ideas generales comunes a todos los problemas de combinación.

Los nombres de los métodos propuestos no fueron tomados de ningún texto, son simplemente nombres que se asignaron a los métodos para hacer referencia en el alcance de este trabajo.

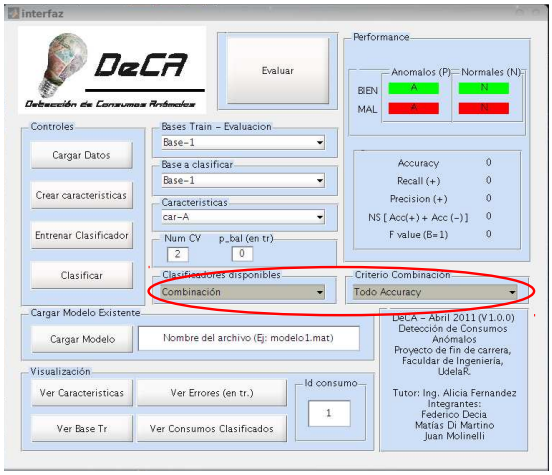
12.1.1. Todo Accuracy

El primer método que se decide implementar, es el esquema de votos por mayoría ponderada descripto en la sección 7.1.1 (pag. 68). Dada una muestra x, calculamos para cada clase:

$$g_p(x) = log\left(P(\omega_p)\right) + \sum_{i=1}^{L} d_p^i \, b_i \quad (12.5)$$

donde

$$b_i \propto log\left(\frac{p_i}{1-p_i}\right) \quad (12.6)$$



Como se puede observar, la decisión tomada por cada clasificador es ponderada por un coeficiente que depende de la probabilidad de acierto de este. En este caso estimamos la probabilidad de acierto de cada clasificador p_i mediante Acc_i descrito en la ecuación (12.1). Como se puede observar en este tipo de esquema, no se toma en cuenta parte de la información disponible (las probabilidades de acierto individuales dadas por los clasificadores C4.5 y CS-SVM). Las probabilidades a priori $P(\omega_p)$ y $P(\omega_n)$ se calculan en base a la cantidad de muestras de cada clase (p o n según corresponda) sobre el total de muestras en la base de entrenamiento.

Este método es muy utilizado cuando se desea combinar clasificadores, y además maximiza el Accuracy de la combinación [23], sin embargo no es el más adecuado para el problema que se pretende abordar. Recordemos que en problemas donde las clases presentan un gran desbalance y además la clase con menor número de representantes es la que presenta mayor importancia (el interes principal de este proyecto es detectar aquellos consumos anómalos) obtener mayor accuracy no implica estar más cerca de la solución óptima. Recordemos el ejemplo (a efectos ilustrativos) que se vio en la sección 7.1.1, si tenemos un problema donde la clase minoritaria representa el $1\,\%$ del total de las muestras, un clasificador que etiqueta a todas las muestras como normales obtendrá una accuracy del $99\,\%$ (y un $F_{value} = 0$). Este clasificador obtendrá un peso mucho mayor que aquel que tiene por ejemplo un $80\,\%$ de accuracy y $F_{value} = 50\,\%$. Sin embargo a los

efectos de detectar muestras de la clase minoritaria es claro que el segundo clasificador es *mucho mejor* que el primero.

Si bien este método no se ajusta al problema que se pretende abordar, se decidió implementarlo en el esquema final de la solución por ser uno de los métodos que cuenta con mayor formalismo a la hora de su deducción y además es un esquema muy utilizado. Aun cuando este esquema de combinación no será el más adecuado, es sumamente interesante incluirlo a los efectos de tener una referencia para poder comparar otros métodos que se proponen e implementan.

Por último, se propone una pequeña modificación a este método antes de pasar a describir los métodos siguientes de combinación implementados. En lugar de utilizar la ecuación:

$$g_p(x) = log\left(P(\omega_p)\right) + \sum_{i=1}^{L} d_p^i \, log\left(\frac{p_i}{1-p_i}\right) \tag{12.7}$$

$$g_n(x) = log\left(P(\omega_n)\right) + \sum_{i=1}^{L} d_n^i \, log\left(\frac{p_i}{1-p_i}\right) \tag{12.8}$$

introducimos pesos a las probabilidades a priori, como manera de dar mayor importancia a la clase minoritaria. Introducimos los coeficientes α_p y α_n de manera que:

$$g_p(x) = \alpha_p \, log\left(P(\omega_p)\right) + \sum_{i=1}^{L} d_p^i \, log\left(\frac{p_i}{1-p_i}\right) \tag{12.9}$$

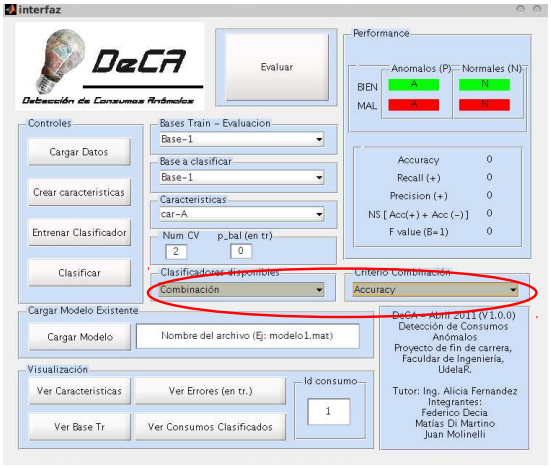
$$g_n(x) = \alpha_n \, log\left(P(\omega_n)\right) + \sum_{i=1}^{L} d_n^i \, log\left(\frac{p_i}{1-p_i}\right) \tag{12.10}$$

De este modo, podemos compensar el desbalance que presentan las clases y otorgar mayor importancia a los consumos anómalos que son aquellos sobre los que se quiere tener mayor capacidad de detección. En particular se ensayaron valores de $\alpha_p = log\left(P(\omega_p)\right)^{-1}$ y $\alpha_n = log\left(P(\omega_n)\right)^{-1}$. Si bien introducir estos valores empeora el accuracy de la combinación (como es de esperarse) mejora el F_{value} (que es el interés principal).

12.1.2. Accuracy

Si bien el método antes mencionado permite obtener una combinación *óptima* en el sentido que maximiza el accuracy de la combinación, está lejos de ser el mejor esquema posible para el problema puntual que se pretende abordar. En particular porque el método de voto por mayoría, tiene cabida fundamentalmente cuando se pretende combinar clasificadores cuyas salidas son únicamente las etiquetas. Cuando se dispone de probabilidades de pertenencia a una u otra clase para **cada muestra**, en general son más populares los esquemas de combinación que utilizan dicha información y delimitan zonas en el espacio de muestras donde cada clasificador es más preciso que sus competidores.

Teniendo en mente lo anterior, se propone modificar el esquema de voto por mayoría, utilizando en lugar de la probabilidad de acierto global calculada para cada clasificador, la probabilidad de pertenencia a la clase asignada para cada muestra en el caso de C4.5 y CS-SVM).



Para ilustrar el nuevo esquema de combinación propuesto, supongamos que queremos clasificar una muestra x, cuyas etiquetas asignadas por los clasificadores base fueron: $(d_p^1(x)\ d_n^1(x))$, $(d_p^2(x)\ d_n^2(x))$, $(d_p^3(x)\ \ d_n^3(x))$ y $(d_p^4(x)\ \ d_n^4(x))$ [b]. La función binaria $d_p^i(x)$ toma el valor 1 si el i-esimo clasificador asigna a x la clase p y $d_p^i(x) = 0$ en otro caso. Analogamente se definen los $d_n^i(x)$ para la clase n. Recordemos que los clasificadores 2 y 3 (CS-SVM y C4.5 respectivamente) también arrojan como salida las probabilidades de pertenencia a una otra clase, $(P_p^2(x)P_n^2(x))$ y $(P_p^3(x)P_n^3(x))$. La nueva estrategia de combinación queda entonces:

$$\begin{aligned} g_p(x) &= \alpha_p\, log\,(P(\omega_p)) + d_p^1(x)\, log\left(\frac{Acc_1}{1-Acc_1}\right) + d_p^2(x)\, log\left(\frac{P_p^2(x)}{1-P_p^2(x)}\right) \\ &\quad + d_p^3(x)\, log\left(\frac{P_p^3(x)}{1-P_p^3(x)}\right) + d_p^4(x)\, log\left(\frac{Acc_4}{1-Acc_4}\right) \end{aligned}$$

$$\begin{aligned} g_n(x) &= \alpha_n\, log\,(P(\omega_n)) + d_n^1(x)\, log\left(\frac{Acc_1}{1-Acc_1}\right) + d_n^2(x)\, log\left(\frac{P_n^2(x)}{1-P_n^2(x)}\right) \\ &\quad + d_n^3(x)\, log\left(\frac{P_n^3(x)}{1-P_n^3(x)}\right) + d_n(x)^4\, log\left(\frac{Acc_4}{1-Acc_4}\right) \end{aligned}$$

Finalmente como es habitual asignamos a la muestra la clase p si $g_p(x) \geq g_n(x)$ y n en caso contrario.

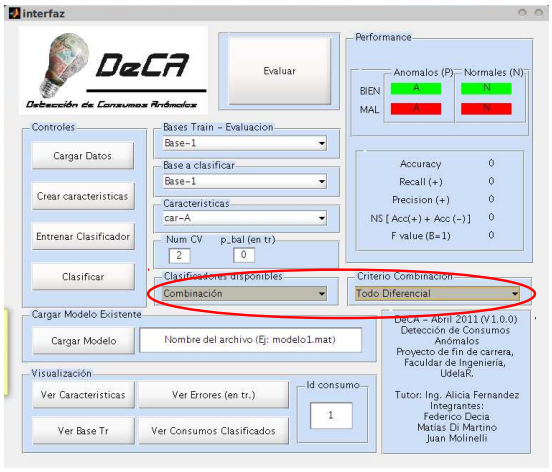
12.1.3. Todo Diferencial

Si bien en el modo ”Accuracy” se aprovecha la información individual que arrojan algunos de los clasificadores, se continua ponderando a los clasificadores en función de su probabilidad de acierto, sin tener en cuenta medidas de performance aptas para problemas con desbalance de clases. Como se analizo en capítulos previos , obtener un clasificador con mejores probabilidades de acierto, no asegura que estamos recorriendo el camino correcto a la hora de la detección de muestras de la clase minoritaria. Como alternativa para solucionar este inconveniente se propone el esquema que se describe a

[b] Hemos asignado arbitrariamente el super indice 1 para referirnos al clasificador One Clas SVM, el 2 para CS-SVM el 3 para el arbol C4.5 con Adaboost y 4 para OPF.

continuación.

Supongamos que clasificamos una base de datos previamente etiquetada utilizando validación cruzada[c], y obtenemos por lo tanto una matriz de confusión que caracteriza el desempeño de dicho clasificador para la base seleccionada. En dicha matriz obtenemos la cantidad de muestras de la clase minoritaria correctamente clasificadas TP, la cantidad de la clase minoritaria mal clasificadas FN las muestras de la clase normal o mayoritaria correctamente clasificadas TN y por último la cantidad de muestras normales mal clasificadas FP. Si recordamos la definición de



$$Acc_p = \frac{TP}{TP + FN} \qquad Acc_n = \frac{TN}{TN + FP}$$

podemos observar que existe una relación entre Acc_p y la cantidad de *falsos normales* (consumos anómalos que se clasifican como normales) así como también una relación similar entre Acc_n y los *falsos anómalos* (consumos normales clasificados como anómalos). Teniendo esto en mente, parece razonable utilizar Acc_p como una medida de la confianza que se tiene sobre una etiqueta normal (asignada por el clasificador). En otras palabras, cuando el clasificador asigne a una muestra dada la etiqueta n (normal o negative), podemos tener en cuenta el valor de Acc_p para medir la *confianza* sobre la decisión tomada. Un valor elevado de Acc_p indica que el número de FN es pequeño (en relación a TP), es decir, tenemos *pocos* falsos negativos lo que se traduce en un índice alto de aciertos sobre las etiquetas normales (en la base de entrenamiento). Análogamente podemos utilizar Acc_n para evaluar la confianza sobre las etiquetas p asignadas (positive o anómalo). De este modo se propone utilizar un esquema que llamamos *diferencial* donde en lugar de utilizar Acc para evaluar la confianza de las etiquetas sin discriminación, se utuliza Acc_p para evaluar la confianza sobre las etiquetas n y Acc_n para evaluar la confianza sobre las etiquetas p.

Definimos entonces

$$g_p(x) = d_p^1(x)\, Acc_n^1 + d_p^2(x)\, Acc_n^2 + d_p^3(x)\, Acc_n^3 + d_p^4(x)\, Acc_n^4 \tag{12.11}$$

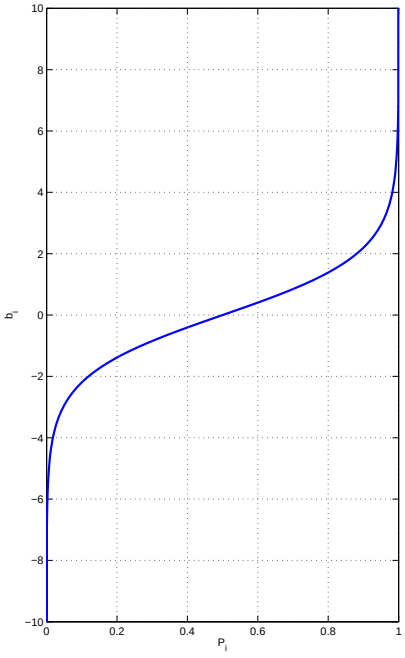
$$g_n(x) = d_n^1(x)\, Acc_p^1 + d_n^2(x)\, Acc_p^2 + d_n^3(x)\, Acc_p^3 + d_n^4(x)\, Acc_p^4 \tag{12.12}$$

De este modo, cuando $g_p(x) \geq g_n(x)$ asignaremos a la muestra x la etiqueta de anómalo y en caso contrario la etiqueta de normal (o negative).

[c]entrenamos los clasificadores con una porción de la base y clasificamos la porción restante. Luego repetimos el proceso rotando los conjuntos que se clasifican hasta tener toda la base de datos clasificada.

Antes de continuar describiendo los siguientes métodos de combinación propuestos, veamos una particularidad de las ecuaciones (12.11) y (12.12) con respecto a (12.9) y (12.10). En primer lugar se elimino el coeficiente correspondiente a las probabilidades a priori ($\log(P(\omega_i))$), esto se debe a fijar $\alpha_n = \log(P(\omega_n))^{-1}$ y $\alpha_p = \log(P(\omega_p))^{-1}$ (luego eliminamos el factor constante 1 ya que solo importa el signo de $g_p(x) - g_n(x)$ a la hora de decidir que clase asignamos a cada muestra). Fijando α_n y α_p de esa manera se contribuye a compensar el desbalance de las clases, si bien el accuracy de la combinación disminuye, el F_{value} aumenta como se puede ver en la presentación de resultados. Otra diferencia que se puede notar entre las expresiones dadas en (12.11) y (12.12) con respecto a (12.9) y (12.10) consiste en la forma en que son ponderadas los Acc en uno y otro método. Recordemos que en [23] se demuestra que la combinación de los clasificadores independientes con probabilidad de acierto individual P_i, maximiza la probabilidad de acierto del conjunto tomando coeficientes,

$$b_i \propto log\left(\frac{p_i}{1-p_i}\right) \tag{12.13}$$



estos coeficientes como se puede observar en la ecuación (12.13) se calculan mediante una operación no lineal sobre las probabilidades de acierto P_i, donde para valores de P_i cercanos a $0{,}5$, los coeficientes tienden a cero y para valores de P_i cercanos a 1 el coeficiente se dispara hacia $+\infty$. Esta transformación es razonable para el caso de probabilidades de acierto de un clasificador. Si la probabilidad de acierto fuera de $0{,}5$ es coherente que el peso del clasificador sea nulo ya que clasifica de igual modo que si asignamos una etiqueta tirando una moneda. Por su parte un clasificador con probabilidad de acierto cercana a la unidad no comete prácticamente errores siendo razonable que el peso del mismo sea alto dentro de la combinación. Los valores que puede tomar P_i siempre están por encima de $0{,}5$ (en caso contrario invirtiendo la salida tenemos un clasificador de mejor performance) razón por la cual los pesos de los clasificadores nunca toman valores negativos.

La ecuación 12.13 se calcula al maximizar la probabilidad de acierto de clasificadores independientes [23], la pregunta que surge naturalmente a estas alturas es ¿por qué no usar la misma transformación dada en 12.13 con Acc_p y Acc_n en (12.11) y (12.12)?

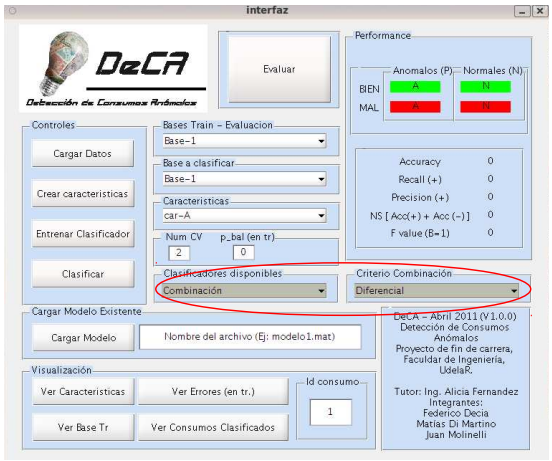
La respuesta a la pregunta no es sencilla (desde el punto de vista teórico), no se ha encontrado en la bibliografía adaptaciones al teorema donde se deducen los coeficientes b_i para el caso en que se desea trabajar con clases desbalanceadas, ni se conoce a priori el impacto de aplicar esta transformación para cantidades que no sean la probabilidad de acierto. Por otro lado observemos que para el caso de Acc_p y Acc_n no nos restringimos al intervalos $[0{,}5\ 1]$ pues en problemas con clases desbalanceadas

como el que se pretende abordar, valores de Acc_p de $0{,}3$ o $0{,}4$ pueden ser razonables. La decisión que se tomó en este aspecto fue de probar ambas estrategias. Se evaluó para una base de datos previamente etiquetada cómo variaba la performance de la combinación (comparando fundamentalmente el F_{value}) utilizando el esquema propuesto en (12.11) y (12.12) contra el esquema equivalente pero asignando los pesos como $\log\left(\frac{Acc_i}{1-Acc_i}\right)$. Los resultados eran los esperados, en el caso de los pesos calculados a partir de (12.13) utilizando como argumentos Acc_i se obtenían valores notoriamente bajos de F_{value} (del orden de $30\,\%$) comparado con el esquema propuesto que presentaba F_{value} del orden de $55\,\%$.

Si bien las observaciones anteriores justifican la implementación del método en la forma dada por la ecuaciones (12.11) y (12.12) (sin aplicar la transformación (12.13)) esto no quiere decir que no exista alguna operación $b_i = f(Acc_i)$ tal que la combinación maximice el F_{value}. De todos modos la deducción teórica de f que permita obtener la combinación óptima (en el contexto en que maximiza el F_{value}) no se encontró en las publicaciones disponibles ni se alcanzó a deducir en este trabajo[d].

12.1.4. Diferencial

Si bien el método introducido en la sección anterior arrojaba buenos resultados y parece ser una alternativa razonable a la hora de tomar especial consideración de los consumos de la clase minoritaria, podemos hacer la misma crítica que se hacía al método ”típico” o más estándar de combinación introducido en la sección 12.1.1. Al considerar indicadores globales de la performance, ¿no se está desperdiciando la información adicional de la confianza sobre cada etiqueta proporcionada por los clasificadores CS-SVM y C4.5 con Adaboost?



Para contemplar la pregunta anterior, se decide implementar un cuarto método de combinación de los clasificadores, en el cual al igual que en el método ”Accuracy” propuesto en 12.1.2 se combine la información global de performance para los clasificadores One Class SVM y OPF (1 y 4 respectivamente) y la información disponible para cada muestra en el caso de CS-SVM y C4.5 con Adaboost (clasidicadores número 2 y 3 respectivamente). De este modo el esquema de combinación propuesto queda de la forma:

$$g_p(x) = d_p^1(x)\, Acc_n^1 + d_p^2(x)\, P_p^2(x) + d_p^3(x)\, P_p^3(x) + d_p^4(x)\, Acc_n^4 \tag{12.14}$$

$$g_n(x) = d_n^1(x)\, Acc_p^1 + d_n^2(x)\, P_n^2(x) + d_n^3(x)\, P_n^3(x) + d_n^4(x)\, Acc_p^4 \tag{12.15}$$

[d] pese a unos tímidos intentos.

Nuevamente no se toman en cuenta transformaciones para obtener coeficientes en función de las P_i o Acc_i a modo de ponderar cantidades comparables (todos multiplicadores en $[0\ 1]$ y no en $[0\ +\infty]$ como en el caso de los b_i).

12.1.5. Iterativo

El último esquema de combinación implementado, es tal vez el más simple de todos los esquemas propuestos. A pesar de lo anterior es uno de los métodos que presenta mejores resultados.

Se define al igual que en los métodos anteriores

$$g_p(x) = d_p^1(x)\, b_1 + d_p^2(x)\, b_2 + d_p^3(x)\, b_3 + d_p^4(x)\, b_4 \tag{12.16}$$

$$g_n(x) = d_n^1(x)\, b_1 + d_n^2(x)\, b_2 + d_n^3(x)\, b_3 + d_n^4(x)\, b_4 \tag{12.17}$$

Si observamos las ecuaciones (12.16) y (12.17) podemos ver que el esquema propuesto tiene la forma de voto ponderado. La variación respecto al método "Todo Accuracy" propuesto en la sección 12.1.1 radica en cómo se obtienen los coeficientes b_i que determinan el peso de cada clasificador en la combinación.

Mientas que en el esquema de voto ponderado *tradicional* los pesos se determinan a partir de las probabilidades de acierto de cada uno de los clasificadores en competencia, en este método se propone calcular los coeficientes b_i que mejoran el valor de F_{value} (en una base de entrenamiento previamente etiquetada). El procedimiento propuesto consiste en definir un conjunto de valores posibles para cada uno de los coeficientes, por ejemplo: $b_i \in [0 : 0{,}05 : 1]$. Luego, para cada una de las posibilidades tenemos definida una regla de combinación (si tomamos por ejemplo 20 valores para cada b_i tenemos 20^4 posibilidades)[e]. Utilizando la misma clasificamos la base de entrenamiento[f] y calculamos el F_{value}. Finalmente definimos como pesos b_i aquellos con los cuales se obtuvo mejor F_{value}.

Si bien a primera vista el método puede parecer muy costoso (por el número de iteraciones) en realidad es un proceso bastante rápido (toma decenas de segundos) en comparación con los tiempos de entrenamiento de los clasificadores (que toman horas).

[e] si bien hay posibilidades iguales dentro de la estrategia propuesta (por ejemplo $b_1 = b_2 = b_3 = b_4 = 0{,}1$, $b_1 = b_2 = b_3 = b_4 = 0{,}2$, etc) no se prestó atención a este hecho ya que los tiempos de ejecución eran cortos y no fue necesario optimizaciones en ese aspecto.

[f] como siempre realizando validación cruzada para eliminar sobre ajustes

Esto se debe a que no es necesario entrenar clasificadores en cada iteración, lo que está cambiando en cada iteracion es el modo de combinar las etiquetas que arroja cada uno de los clasificadores individuales (pero no es necesario generar las etiquetas individuales cada vez). Por este motivo en cada iteración simplemente se realizan un par de decenas de operaciones.

Este método es uno de los pocos que se encontró en publicaciones que atacan problemas particulares de combinación de clasificadores con clases desbalanceadas [3]. Sin embargo el método propuesto por Nitesh et al. [3] utiliza un algoritmo de aprendizaje genético (GA) para encontrar una solución sub-óptima. A pesar de esa diferencia la base del método coincide ya que se ensayan varias combinaciones de pesos, se calcula el F_{value} para cada combinación y se toma el set de valores que arrojaron mejor F_{value}. En este trabajo no se consideraron algoritmos de búsqueda más rápidos que la búsqueda exhaustiva pues el costo computacional que demanda el esquema propuesto es muy bajo en comparación con los costos promedio que se manejan en el entrenamiento de los clasificadores.

CAPÍTULO 13

Resultados

Este capítulo presenta los resultados obtenidos en dos niveles distintos. En una primera parte, se presentarán los resultados para las distintas configuraciones propuestas. Se evaluarán entre otras cosas los clasificadores propuestos, los modos de combinación y el impacto de balancear la base de entrenamiento. Todas las conclusiones teóricas serán realizadas en base a la performance obtenida para la base-2, comparando los resultados que arroja cada estrategia con las etiquetas disponibles. Luego, utilizaremos la Base-3 para realizar una evaluación de resultados en campo como se explicará a continuación.

La segunda parte del capítulo, presenta los resultados obtenidos *en campo*. Una vez definidas las estrategias más promisorias, se clasificó una base independiente (base-3) provista por UTE. Esta nueva base fue etiquetada en una primera instancia para luego realizar inspecciones de aquellos consumos seleccionados como anómalos. Se presentarán los resultados de dichas inspecciones y se contrastarán con los resultados que se obtienen inspeccionando los consumos seleccionados de manera *manual* por parte del personal experto de UTE.

13.1. Resultados Teóricos

13.1.1. Clasificadores Base

One Class SVM

Comencemos describiendo la performance del clasificador One Class SVM. Se buscó el valor de ν y γ con mejor F_{value} en la clasificación como se describió en el capítulo 11. El intervalo de ν y γ sobre el que se trabajó finalmente es: `r=[-6:1:-1]; nus=[1*10.^r 3*10.^r 5*10.^r 7*10.^r 9*10.^r 1]; gammas=[1*10.^r 3*10.^r 5*10.^r 7*10.^r 9*10.^r];`. Recordemos que las características finales seleccionadas para este algoritmo son $[1,\ 2,\ 3,\ 12,\ 20,\ 28]$ como se mostró en la sección 10.3.

Los resultados obtenidos son:

- Accuracy=84,9 %
- Recall=54,9 %
- Precision=50,8 %
- F-value=52,8 %

CS-SVM

El clasificador C-SVM sensible al costo, fue uno de los clasificadores que arrojó mejores resultados. Los rangos finales de C y γ sobre los que se realiza la búsqueda son:

```
e = [-4:1:1]; vec_C = [1*10.^e 3*10.^e 5*10.^e 7*10.^e 9*10.^e 1];
r = [-7:1:-1]; vec_gamma = [1*10.^r 3*10.^r 5*10.^r 7*10.^r 9*10.^r];
```

Los pesos asignados a cada una de las clases son:

```
weight_anom = (cant_train/cant_etiq_anom_train)*100;
weight_norm = (cant_train/cant_etiq_norm_train)*100;
```

Las características finales que se utilizaron para este algoritmo son $[1,\ 2,\ 3,\ 11,\ 20,\ 21,\ 23,\ 24,\ 28]$ como se estableció en la sección 10.3.

Los resultados obtenidos son:

- Accuracy=84,5 %
- Recall=62,8 %
- Precision=49,7 %
- F-value=55,5 %

OPF

El tercer clasificador a evaluar es el OPF. En la sección 11.4.2 se analizó el impacto sobre el F_{value} que tiene balancear la base de entrenamiento y se observó que los mejores desempeños se encontraban para valores de p_bal de 0,4, o sea con la base de entrenamiento con 40 % de muestras anómalas. Igualmente, se decidió además analizar el desempeño de este clasificador sin balancear por considerarlo de utilidad a la hora de la combinación. Recordemos que las características finales que se seleccionaron (en la sección 10.3) son los propios valores de los consumos.

Los resultados obtenidos **sin balancear (p_bal=0)** son:

- Accuracy=84,4 %
- Recall=39,6 %
- Precision=49,2 %
- F-value=43,9 %

Los resultados obtenidos **balanceando (p_bal=0.4)** son:

- Accuracy=80,1 %
- Recall=62,2 %
- Precision=40,5 %
- F-value=49,0 %

Como se analizó en capítulos anteriores, balanceando la base de datos de entrenamiento, podemos mejorar la performance del clasificador. Si bien el accuracy baja, el aumento en la detección de consumos anómalos mejora considerablemente (manifestado en el incremento importante del Recall).

Tree (C4.5 con Adaboost)

Al igual que en el caso de OPF, podemos compensar el desbalance de clases y priorizar a las muestras de la clase minoritaria mediante un entrenamiento balanceado. Por esta razón, a continuación se muestras los resultados finales para el Árbol implementado balanceando y sin balancear. Recordemos que las características finales seleccionadas para el Árbol en la sección 10.3 son las *car-A* (características propuestas en [10]). En la versión final propuesta de este clasificador se combinan 15 instancias del árbol C4.5 mediante Adaboost.

Los resultados obtenidos **sin balancear (p_bal=0)** son:

- Accuracy=85,8 %
- Recall=40,2 %
- Precision=55,5 %
- F-value=46,6 %

Los resultados obtenidos **balanceando (p_bal=0.4)** son:

- Accuracy=79,0 %
- Recall=64,6 %
- Precision=39,0 %
- F-value=48,6 %

Al igual que en el caso de OPF, balanceando durante el entrenamiento mejoramos la cantidad de consumos anómalos bien clasificados (Recall). Sin embargo en este caso vemos una disminución importante en la Precision, por esta razón el F_{value} no mejora de manera significativa al balancear la base de entrenamiento.

13.1.2. Clasificadores Combinados

En esta sección mostraremos los resultados obtenidos al combinar mediante las distintas técnicas propuestas en el capítulo 12 los clasificadores base. Estudiaremos el desempeño de combinar a OPF y el Árbol en sus versiones *balanceando* y *sin balancear*.

Combinación *Accuray*

Combinando los clasificadores mostrados anteriormente mediante el método *Accuray* se obtuvo:

Sin balancear OPF y el Árbol (p_bal=0)

- Accuracy=81,3 %
- Recall=43,9 %
- Precision=40,2 %
- F-value=41,9 %

Balanceando OPF y el Árbol (p_bal=0.4)

- Accuracy=86,7 %
- Recall=54,3 %
- Precision=57,0 %
- F-value=55,6 %

Los resultados obtenidos balanceando la base para OPF y el Árbol son mejores que en el caso en que se entrena con el desbalance.

Si observamos el caso balanceado vemos que mejora la mejor de las Accuracy individuales de los clasificadores y tiene prácticamente el F_{value} del mejor de los clasificadores combinados. Se obtiene además un buen compromiso entre el $Recall$ y $Precision$ consiguiendo que ambos indicadores se encuentren por encima del 50 %.

Combinación *Diferecial*

Combinando los clasificadores mostrados anteriormente mediante el método *Diferencial* se obtuvo:

Sin balancear OPF y el Árbol (p_bal=0)

- Accuracy=79,3 %
- Recall=60,3 %
- Precision=39,6 %
- F-value=47,8 %

Balanceando OPF y el Árbol (p_bal=0.4)

- Accuracy=84,3 %
- Recall=62,8 %
- Precision=49,2 %
- F-value=55,2 %

Los resultados obtenidos balanceando la base para OPF y el Árbol son mejores que en el caso en que se entrena con el desbalance.

Si observamos el caso balanceado vemos que el Accuracy es similar a la mejor de las Accuracy individuales de los clasificadores y tiene prácticamente el F_{value} del mejor de los clasificadores combinados. Este método de combinación tiene peor performance que el estudiado antes, presenta peor Accuracy y peor F_{value}. Si bien se obtiene un buen $Recall$ la disminución en la $Precision$ se manifiesta en una baja performance desde el punto de vista del F_{value}.

Combinación *Todo Accuracy*

Combinando los clasificadores mostrados anteriormente mediante el método *Todo Accuracy* se obtuvo:

Sin balancear OPF y el Árbol (p_bal=0)

- Accuracy=87,1 %
- Recall=54,8 %
- Precision=58,8 %
- F-value=56,8 %

Balanceando OPF y el Árbol (p_bal=0.4)

- Accuracy=86,5 %
- Recall=58,5 %
- Precision=55,8 %
- F-value=57,1 %

Para esta estrategia de combinación, se puede observar que la brecha entre balancear o no OPF y el Árbol es menor. Además el Accuracy de la combinación obtenido con este método es superior al de los casos anteriores. Por otro lado el F_{value} de la combinación también es superior a las estrategias mostradas anteriormente.

Combinación *Todo Diferencial*

Combinando los clasificadores mostrados anteriormente mediante el método *Todo Diferencial* se obtuvo:

Sin balancear OPF y el Árbol (p_bal=0)

- Accuracy=86,8 %
- Recall=57,9 %
- Precision=57,3 %
- F-value=57,6 %

Balanceando OPF y el Árbol (p_bal=0.4)

- Accuracy=83,6 %
- Recall=69,5 %
- Precision=47,7 %
- F-value=56,6 %

El desempeño de esta estrategia es similar al de la estrategia *Todo Accuracy* si no balanceamos OPF y el Árbol. En ese caso esta estrategia presenta mejor F_{value} y un poco peor Accuracy. El resultado es el esperado, recordemos que el método "Todo Accuracy", es el que más se asemeja (de todos los métodos propuestos) al modo *canónico* de asignar pesos para maximizar la probabilidad de acierto de la combinación, tal como se demuestra en [23], donde el peso de cada clasificador se asigna en función de la probabilidad de acierto individual más un factor que toma en cuenta las probabilidades a priori de cada clase.

Con OPF y el Árbol entrenados con clases balanceadas, el desempeño de esta estrategia se comporta distinto que la anterior. Se obtiene en este caso un $Recall$ muy alto (del orden del 70 %) y una $Precision$ del 48 % (aprox).

En el caso balanceado, el Accuracy no mejora al del mejor de los clasificadores que se esta combinando, sin embargo en cualquiera de los dos casos el F_{value} supera al de los clasificadores que se están combinando.

Combinación *Iterativo*

Combinando los clasificadores mostrados anteriormente mediante el método *Iterativo* se obtuvo:

Sin balancear OPF y el Árbol (p_bal=0)

- Accuracy=86,9 %
- Recall=57,3 %

- Precision=57,6 %
- F-value=57,5 %

Balanceando OPF y el Árbol (p_bal=0.4)

- Accuracy=86,2 %
- Recall=64,0 %
- Precision=54,4 %
- F-value=58,8 %

Este método es el que obtiene mejor performance (teniendo en cuenta el F_{value}). El Accuracy supera a la de los clasificadores que se están combinando al igual que el F_{value}.

Al igual que sucede en estrategias anteriores, al balancear las bases de entrenamiento para OPF y el Árbol aumenta el $Recall$ (sacrificando un poco la $Precision$).

13.1.3. Análisis de los Resultados

A continuación presentamos en la tabla 13.1 los resultados introducidos en la sección anterior y analizamos los mismos.

Si analizamos los resultados obtenidos para los distintos clasificadores propuestos, podemos observar que CS-SVM presenta el mejor F_{value} seguido por One Class SVM. Para el caso de OPF y el Árbol, se obtienen peores resultados pero estos mejoran al entrenar con bases balanceadas.

Tanto OPF como el Árbol aumentan su $Recall$ al balancear la base de entrenamiento como es de esperar. Sin embargo, también aumenta el número de "falsos anómalos" (FP) produciendo una disminución en la $Precision$. El compromiso óptimo en este caso se encontró para valores de p_bal=0,4.

En cuanto a las estrategias de combinación de clasificadores propuestas, la amplia mayoría mejora la performance del mejor de los clasificadores que se está combinando. Sin embargo se observa mejores resultados para los métodos "Todo Accuracy", "Todo Diferencial" e "Iterativo". Los métodos que consideran las probabilidades de acierto individuales parecen tener peores desempeños a la hora de la combinación.

Entre los métodos de combinación "Todo Accuracy", "Todo Diferencial" e "Iterativo" no se encuentra diferencias apreciables en cuanto a la performance. Cualquiera de las 3 soluciones es una buena estrategia a la hora de combinar los clasificadores. De los resultados obtenidos con p_bal=0 el mejor resultado se encuentra para el esquema "Todo Diferencial" (con comportamientos casi idénticos al modo "Iterativo"). Por otro lado, con p_bal=0.4 se obtiene el mejor resultado de todas las pruebas realizadas, y éste se alcanza

Descripción	Acc.(%)	Rec.(%)	Pre.(%)	F-value(%)
One Class SVM (p_bal=0)	84,9	54,9	50,8	52,8
CS-SVM (p_bal=0)	84,5	62,8	49,7	55,5
OPF (p_bal=0)	84,4	39,6	49,2	43,9
OPF (p_bal=0,4)	80,1	62,2	40,5	49
Tree (C4.5) (p_bal=0)	85,8	40,2	55,5	46,6
Tree (C4.5) (p_bal=0,4)	79	64,6	39	48,6
Combinación “Accuracy” (p_bal=0)	81,3	43,9	40,2	41,9
Combinación “Accuracy” (p_bal=0,4)	86,7	54,3	57	55,6
Combinación “Diferencial” (p_bal=0)	79,3	60,3	39,6	47,8
Combinación “Diferencial” (p_bal=0,4)	84,3	62,8	49,2	55,2
Combinación “Todo Accuracy” (p_bal=0)	87,1	54,8	58,8	56,8
Combinación “Todo Accuracy” (p_bal=0,4)	86,5	58,5	55,8	57,1
Combinación “Todo Diferencial” (p_bal=0)	86,8	57,9	57,3	57,6
Combinación “Todo Diferencial” (p_bal=0,4)	83,6	69,5	47,7	56,6
Combinación “Iterativo” (p_bal=0)	86,9	57,3	57,6	57,5
Combinación “Iterativo” (p_bal=0,4)	86,2	64	54,4	58,8

Cuadro 13.1: Resultados Obtenidos Para la Base-2

utilizando el método de combinación “Iterativo” con un F_{value} del 58,8 %.

Tengamos en cuenta que si bien los métodos “Todo Diferencial” e “Iterativo” presentan muy buenos resultados, el segundo puede estar sobre-ajustado a la base que se utiliza en el entrenamiento. Recordemos que el peso de cada clasificador se obtenía probando con todas las combinaciones y seleccionando aquel juego de parámetros que mejora la performance para **la base de entrenamiento utilizada**. Además se debe tener en cuenta que si bien los tres métodos presentan similares F_{value}, en algunos casos el compromiso entre $Recall$ y $Precision$ varia de un método al otro. En la figura 13.1 se observa el *punto de funcionamiento* que presenta cada uno de los clasificadores donde se puede observar de manera gráfica las variaciones entre el $Recall$ y $Precision$ para los distintos esquemas propuestos. Recordemos que se trabajo con $\beta = 1$, modificando dicho parámetro es posible mover los puntos de la curva en función de las necesidades especificas y la zona de operación deseada.

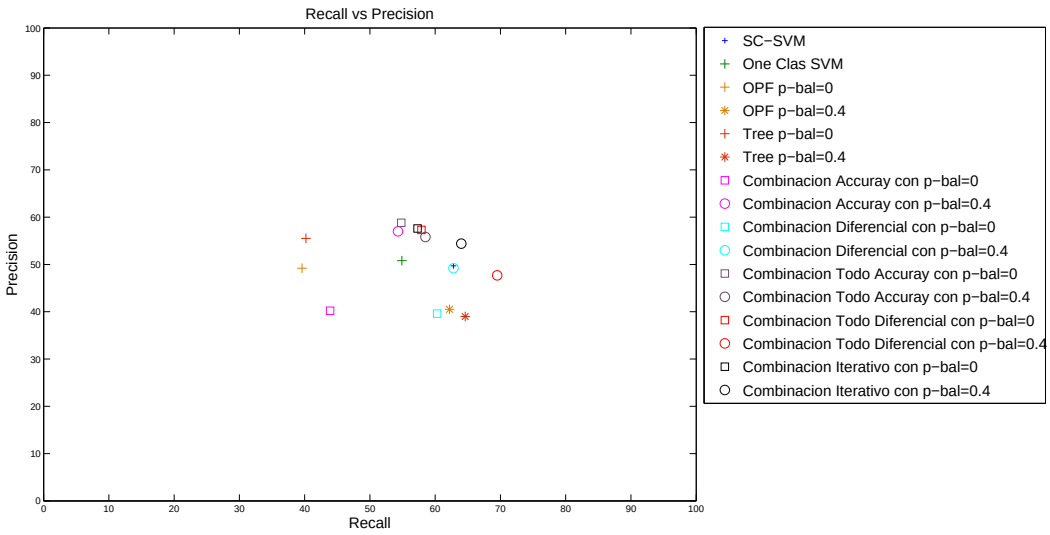


Figura 13.1: $Recall$ y $Precision$ para los distintos clasificadores y modos de combinación implementados.

13.2. Resultados En Campo

13.2.1. Procedimiento

Finalizado el análisis teórico de los resultados, y consolidadas las distintas estrategias que se proponen para atacar el problema, es momento de poner a prueba el sistema. Esto se lleva a cabo etiquetando una nueva base de datos y realizando inspecciones de aquellos clientes que presentan consumos clasificados como anómalos.

Para realizar el procedimiento anterior, se solicitó una nueva base de datos a UTE, llamada en este documento como Base-3. Ésta consta de 3000 consumos (aproximadamente) de clientes comerciales. Los consumos obtenidos son desde enero del 2009 hasta enero del 2011.

El procedimiento realizado para clasificar dicha base de datos fue el siguiente.

1. Se entrenaron 2 clasificadores con la Base-2, utilizando los siguientes parámetros:
   Clasificador A:

   - Combinación - Todo Diferencial
   - p_bal=0

   Clasificador B:

   - Combinación - Todo Diferencial
   - p_bal=0,4

   Las razones para utilizar el modo de combinación “Todo Diferencial” son que este método es el que presenta los mejores resultados junto con el “Iterativo”. Por otro

lado, en contrapartida con este último, el primero tiene menos probabilidades de estar sobre-adaptado a la base de entrenamiento. Lo expresado anteriormente indica que es una estrategia razonable utilizar esta configuración para la clasificación que será contrastada en campo.

2. Se clasificó la Base-3

   Una vez que se entrenó los clasificadores A y B, se clasificó la Base-3 obteniendo los consumos detectados como anómalos por cada uno de los clasificadores. Llamemos Base-A al conjunto de consumos clasificado como anómalos por el clasificador A y Base-B al conjunto clasificado como anómalo por el clasificador B. Como era de esperar, la Base-B tiene más elementos (casi el doble) que la Base-A, al igual que sucedía durante los análisis teóricos. En la tabla 13.1 se observa que cuando balanceamos OPF y el Árbol (p_bal=0,4), existe un aumento en el Recall y un decaimiento en la Precision en el resultado total de la clasificación, lo cual refleja un aumento de la cantidad de consumos clasificados como anómalos . Por otra parte la Base-B incluye casi en su totalidad a la Base-A, lo cual también es de esperar y da la noción de una cierta coherencia, ya que no sería razonable que al balancear la base de entrenamiento en OPF y el Árbol, cambiaran radicalmente las etiquetas que se daban anómalas sin balanceo.

3. Se inspeccionaron los clientes incluidos en Base-A y Base-B.
   Los resultados obtenidos luego de las inspecciones fueron los siguientes:
   Para la Base-A

   - Cantidad de consumos: 265
   - Cantidad de consumos que se inspeccionaron: 200
   - Cantidad de irregularidades detectadas: 6 (3 % de 200)
   - Cantidad de contadores que se enviaron al laboratorio para inspeccionar[a]:4

   Para la Base-B

   - Cantidad de consumos: 560
   - Cantidad de consumos que se inspeccionaron: 340
   - Cantidad de irregularidades detectadas: 11 (3,3 % de 340)
   - Cantidad de contadores que se enviaron al laboratorio para inspeccionar:4

A continuación mostramos aquellos consumos a los que se les detecto irregularidades y luego se analizarán los resultados.

[a]algunos contadores se enviaron al laboratorio de UTE pues presentaban características sospechosas. Estos casos se pueden sumar al número de irregularidades detectadas en función del resultado que se obtenga del laboratorio

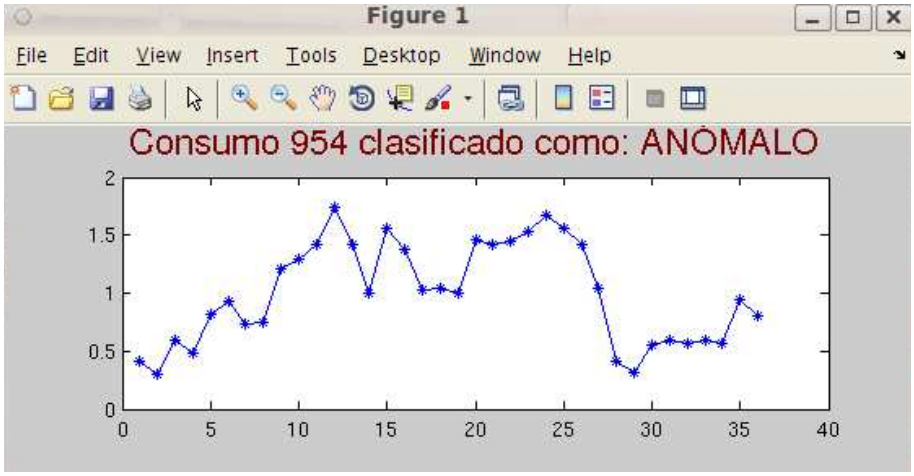


Figure 1
File Edit View Insert Tools Desktop Window Help
Consumo 954 clasificado como: ANÓMALO
2
1.5
1
0.5
0
0 5 10 15 20 25 30 35 40


Figure 1
File Edit View Insert Tools Desktop Window Help
Consumo 1361 clasificado como: ANÓMALO
2
1.5
1
0.5
0
0 5 10 15 20 25 30 35 40


Figure 1
File Edit View Insert Tools Desktop Window Help
Consumo 1775 clasificado como: ANÓMALO
Consumo con irregularidad detectada el: 141206
3
2
1
0
0 5 10 15 20 25 30 35 40


Figure 1
File Edit View Insert Tools Desktop Window Help
Consumo 1926 clasificado como: ANÓMALO


Figure 1
File Edit View Insert Tools Desktop Window Help
Consumo 2537 clasificado como: ANÓMALO
Consumo con irregularidad detectada el: 140208


Figure 1
File Edit View Insert Tools Desktop Window Help
Consumo 2226 clasificado como: ANÓMALO
Consumo con irregularidad detectada el: 111007

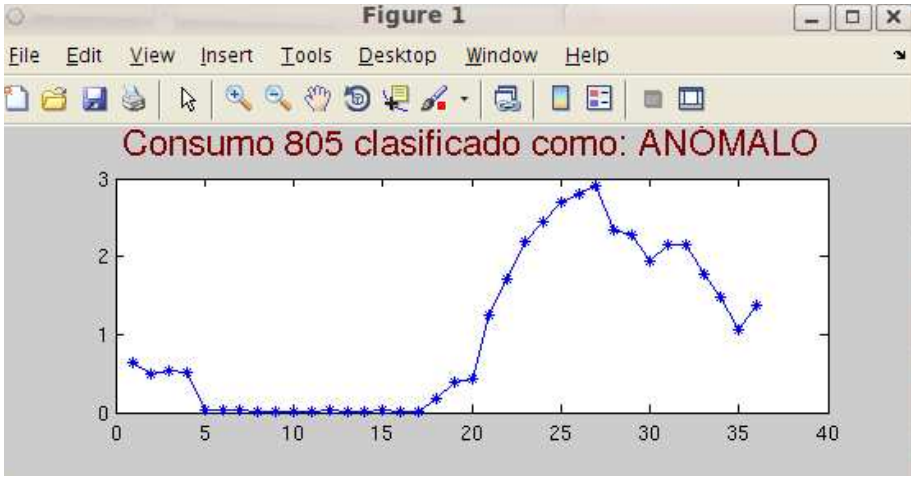


Figure 1
File Edit View Insert Tools Desktop Window Help
Consumo 805 clasificado como: ANOMALO
0 1 2 3
0 5 10 15 20 25 30 35 40


Figure 1
File Edit View Insert Tools Desktop Window Help
Consumo 341 clasificado como: ANOMALO
0 0.5 1 1.5 2
0 5 10 15 20 25 30 35 40


Figure 1
File Edit View Insert Tools Desktop Window Help
Consumo 382 clasificado como: ANOMALO
0 0.5 1 1.5 2
0 5 10 15 20 25 30 35 40

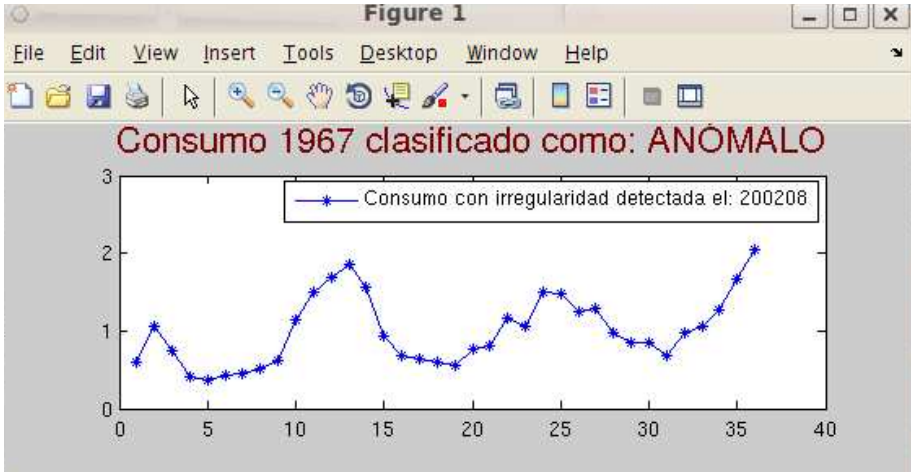


Figure 1
File Edit View Insert Tools Desktop Window Help
Consumo 1967 clasificado como: ANÓMALO
Consumo con irregularidad detectada el: 200208
0 1 2 3
0 5 10 15 20 25 30 35 40


Figure 1
File Edit View Insert Tools Desktop Window Help
Consumo 1146 clasificado como: ANÓMALO
0 0.5 1 1.5 2
0 5 10 15 20 25 30 35 40

13.2.2. Análisis de los resultados

Para analizar los resultados obtenidos, en primer lugar debemos realizar precisiones sobre el procedimiento realizado por UTE para la identificación de fraudes. De este modo, tendremos una noción de qué valores numéricos representan *buenos* resultados y cuáles representan *malos* resultados.

Para ubicar los resultados obtenidos en la práctica, analicemos algunos valores.**La cantidad de clientes consumidores de energía eléctrica en Montevideo ronda el medio millón**. **De estos aproximadamente 30.000 clientes son los llamados *clientes estratégicos***. Bajo esta denominación se incluyen en general clientes comerciales y algunos hogares con alta potencia contratada, que el departamento de detección de irregularidades de UTE controla con cierta periodicidad. Algunos clientes pueden entrar al conjunto de *clientes estratégicos* si se cree necesario, o vice versa, clientes estratégicos pueden abandonar esta categoría. A pesar de esto se puede decir que en líneas generales el conjunto de clientes estratégicos se mantiene bastante estático desde hace varios años.

De los 30.000 clientes estratégicos aproximadamente 9.000 clientes por año son seleccionados mediante diferentes técnicas y reglas para ser inspeccionados. Un cliente puede ser meritorio de inspección por varias razones, como puede ser una denuncia, por presentar bajos consumos para la potencia contratada, por tener poca potencia contratada en función de los metros cuadrados que tiene el local o por tener antecedentes de fraude, entre otros factores que son tenidos en cuenta. Los criterios a la hora de seleccionar consumos a inspeccionar son muy variados.

De los 9.000 clientes que se inspeccionan por año (aproximadamente), se detectó en el 2008 que un 7 % presentaban irregularidades, en el 2009 un 5 % y en el 2010 un 4 %. Esto quiere decir que, por ejemplo, en el 2010 de las 9.000 inspecciones realizadas, el 4 % (360 clientes) tenían irregularidades de algún tipo. Estas irregularidades pueden ser un fraude por una manipulación del contador o la acometida por ejemplo, o puede tratarse simplemente de un contador defectuoso. Si observamos el porcentaje de *efectividad* que tienen las inspecciones, podemos ver que viene decreciendo al pasar los años (7 % en el 2008, 5 % en 2009 y 4 % durante 2010), lo que indica una disminución en la cantidad de clientes fraudulentos detectados dentro de la base de *clientes estratégicos*. Esta disminución, según los técnicos de UTE, se debería a que durante los últimos años se viene controlando al mismo subconjunto de clientes, reinstalando muchos contadores fuera de los recintos donde son más visibles, mejorando las conexiones con la acometida y utilizando contadores digitales. Por otro lado, las multas y sanciones aplicadas a aquellos clientes que cometen fraude, o tan solo el hecho de estar siendo controlado, han ido aplacando las intenciones de cometer ilícitos dentro de estos clientes.

Teniendo en mente las puntualizaciones anteriores, analicemos los resultados arrojados por las inspecciones. En primer lugar, calculemos la cantidad de consumos anómalos *presentes en la Base-3* (base de consumos clasificados). Teniendo en cuenta que de los 30.000 clientes estratégicos se inspeccionaron 9.000 durante el 2010 y de éstos el 4 % presentaba irregularidades, y además, asumiendo como hipótesis que el procedimiento manual es exhaustivo y exitoso, podemos estimar la densidad de clientes

fraudulentos como:

$$\text{Densidad de Clientes Fraudulentos} = \frac{9000 \times 0{,}04}{30000} = 1{,}2\,\%$$

Podemos utilizar el resultado anterior, para estimar la cantidad de clientes fraudulentos dentro de la Base-3 (base con 3.000 consumos de clientes estratégicos) como:

$$\text{Clientes Fraudulentos en Base-3} = 0{,}012 \times 3000 = 36$$

Por otro lado, de los 560 consumos que se mandaron inspeccionar, por razones de tiempo, se inspeccionaron a la fecha 340 consumos, detectando en total 11 irregularidades y 4 situaciones sospechosas que están siento estudiadas en el laboratorio. Esto arroja una cantidad de clientes fraudulentos detectados de 11 en el caso en todos los ensayos de laboratorio den negativos y 15 en el caso opuesto. Con estos resultados podemos estimar[b] que la cantidad de clientes fraudulentos detectados se encuentra dentro del intervalo $\left[11 \times \frac{560}{340}, 15 \times \frac{560}{340}\right] \approx [18, 25]$. En conclusión la cantidad de clientes fraudulentos detectados se encuentra entre el $50\,\%$ (18 de 36) y el $69\,\%$ (25 de 36) de los clientes presentes en la Base-3.

También podemos hacer una interpretación más directa de los resultados, comparando el porcentaje de clientes fraudulentos en el conjunto de clientes inspeccionados. De las inspecciones realizadas, entre el 3 y 4 %[c] de los clientes cometían fraude. Este resultado es comparable a los resultados obtenidos por los expertos de UTE. Debe tenerse en consideración además, que la cantidad de recursos utilizados en la clasificación automática es prácticamente nula.

En resumen: se considera que las pruebas realizadas sustentan la utilidad de la herramienta propuesta para asistir al personal de UTE para programar las inspecciones. En la prueba realizada se hicieron inspecciones sin que mediara ningún análisis de los registros por el personal especializado y por los tanto sin utilizar ninguna hora hombre en la tarea de selección. Por lo que la obtención de porcentajes de éxito por inspección, similares a los que se obtiene en el procedimiento manual es auspicioso, aún más cuando se hizo basado exclusivamente con la información de los registros de consumos sin tener en cuenta por ejemplo los antecedentes de fraude.

[b] los resultados pueden ser incluso mejores que los que se aquí se estiman, pues las inspecciones se comienzan a realizar desde las zonas más cercanas a UTE hacia las más alejadas. En las zonas más alejadas es donde se detecta mayor índices de fraude. Por este motivo se espera que para los consumos que faltan inspeccionar el índice detección sea un poco más alto que el actual.

[c] recordemos que aun está pendiente el resultado de los contadores enviados al laboratorio

CAPÍTULO 14

Conclusiones y trabajos a futuro

14.1. Conclusiones

Una vez que se ha ahondado en la descripción del problema, las herramientas teóricas que utilizamos, su implementación y los resultados obtenidos, en esta sección extraeremos algunas conclusiones de lo que fue nuestro trabajo.

14.1.1. Marco Teórico

Consideramos que el presente trabajo se enmarcó en la teoría de reconocimiento de patrones y su aplicación en la detección de fraudes en el consumo eléctrico. Se tuvieron en cuenta los estudios que actualmente se vienen realizando en la búsqueda de algoritmos que detecten consumos sospechosos de fraude, como un punto de partida, sumándole a esta experiencia la particularidades de nuestro problema. A medida que fuimos avanzando identificamos problemas conocidos en la literatura de la materia, de modo que acudiendo a las publicaciones adecuadas pudimos cimentar nuestras decisiones en una base teórica firme. Un ejemplo es el tratamiento que se debe dar a problemas donde las clases se encuentran notoriamente desbalanceadas, que incluye desde la modificación de la distribuciones de las clases, por ejemplo a partir del submuestreo, hasta la inclusión de medidas de performance como el F_{value}. Tanto en éste como en otros temas se puede observar referencias a libros y publicaciones que fundamentan las opciones tomadas.

14.1.2. Resultados

A la hora de realizar una conclusión de los resultados obtenidos con la herramienta desarrollada, debemos observar dos aspectos, por un lado los resultados teóricos y por otro los resultados en campo que derivaron de las inspecciones de los consumos seleccionados.

En lo que refiere a los resultados teóricos, se valoran como satisfactorios. Observamos como a lo largo del camino recorrido durante el proyecto se fue mejorando el

desempeño de manera progresiva. En capítulos anteriores destacamos como los métodos de selección de características y combinación de clasificadores implementados influyeron positivamente en el desempeño final. Por otro lado, no existen valores de performance claros, publicados en la literatura, que nos permitan realizar una comparación de los resultados teóricos. Sin embargo, si comparamos los resultados obtenidos con los resultados intermedios que se obtenían a lo largo del transcurso del tiempo, vemos de manera satisfactoria como fue mejorando el desempeño del esquema de solución propuesto. Por otro lado pudimos validar el trabajo realizado poniendo a prueba el sistema implementado mediante inspecciones en campo con los valores arrojados.

En cuanto a los resultados en campo, como se expuso en el capítulo 13, donde se abordaron los resultados de las inspecciones, podemos decir que son muy satisfactorios. La efectividad de las inspecciones es similar a las que obtienen los técnicos de UTE realizando el trabajo manual. Nada indicaba que nuestra predicción no pudiese ser mejor que la de los expertos, pero teniendo en cuenta que el entrenamiento fue realizado en base a su conocimiento, el tener resultados similares es muy alentador.

14.1.3. Software

Se cuenta con un software que posee una interfaz de usuario amigable que permite ejecutar los algoritmos de manera sencilla. Es claro que todavía queda camino para recorrer en este sentido, pero creemos que el programa actual posee un código interpretable, que con un poco de estudio no será difícil de modificar para introducir futuras mejoras

14.1.4. Gestión de proyecto

En esta subsección se realizará una evaluación de la gestión inicial del proyecto. Compararemos los tiempos planeados inicialmente para cada una de las tareas con el que insumió cada una de ellas a lo largo de este proyecto. En la imagen 14.1 se puede observar el plan inicial del proyecto, el cual muestra para cada una de las tareas el tiempo en el cual se pretendía concretarlas

La duración de las tareas difirió de lo planeado por varios aspectos,

- Hubo un mayor tiempo que el esperado al comienzo del proyecto para familiarizarse con el problema. Esto incluye el proyecto anterior de la materia reconocimiento de patrones que abarcó este tema en una primera instancia y también lo que es la materia prima del trabajo, los consumos. Hubo que adquirir algo del conocimiento de los expertos de UTE acerca del etiquetado de los mismos y a su vez preparar la base de datos para que sea la entrada a nuestro sistema.

- Relacionado con lo anterior, se subestimó el tiempo de estudio teórico del problema. Fue importante en cada parte hacer una búsqueda de lo que actualmente está publicado sobre el tema y ver si se podía adaptar a nuestro problema.

- También hubo un cambio importante en el orden en que se abordaron los temas. Inicialmente el proyecto estaba pensado de una forma muy lineal, realizando cada bloque en el orden en que estaban planteados en el diagrama de bloques (ver figura

Nombre	Duración	Inicio	Terminado
Generacion Base de Datos	**30 days**	**10/05/10 08:00 AM**	**18/06/10 05:00 PM**
Pre-Procesamiento	30 days	10/05/10 08:00 AM	18/06/10 05:00 PM
Inspeccion Caracteristicas	30 days	10/05/10 08:00 AM	18/06/10 05:00 PM
Desarrollo Algoritmos de Reconocimien...	**105 days**	**21/06/10 08:00 AM**	**12/11/10 05:00 PM**
Seleccion/Extraccion de Caracteristicas	30 days	21/06/10 08:00 AM	30/07/10 05:00 PM
SeleccionClasificador/es	45 days	02/08/10 08:00 AM	01/10/10 05:00 PM
Entrenamiento y Evaluacion	30 days	04/10/10 08:00 AM	12/11/10 05:00 PM
Generacion Base de Datos 2	20 days	15/07/10 07:00 AM	11/08/10 05:00 PM
Estudio del Desempeno en Base de Datos 2	15 days	15/11/10 08:00 AM	03/12/10 05:00 PM
Documentacion	20 days	06/12/10 08:00 AM	31/12/10 05:00 PM
Primer presentación	0 days	15/09/10 08:00 AM	15/09/10 08:00 AM
Preparación de la primer presentacion	3 days	10/09/10 08:00 AM	14/09/10 05:00 PM
Segunda presentacion	0 days	15/02/11 05:00 PM	15/02/11 05:00 PM
Preparacion segunda presentacion	4 days	10/02/11 08:00 AM	15/02/11 05:00 PM
Presentacion final	1 day	01/03/11 08:00 AM	01/03/11 05:00 PM
Preparacion de la presentacion	7 days	15/02/11 08:00 AM	23/02/11 05:00 PM

Figura 14.1: Imagen que muestra los tiempos de cada tarea en el plan inicial de proyecto

2.1 en la página 18). En determinado momento resolvimos que era mejor tener primero un prototipo del sistema funcionando y luego ir mejorando el conjunto a medida que se trabajaba en cada bloque. Esto nos brindó la ventaja de tener un constante feedback mientras introducíamos modificaciones.

14.2. Trabajo a futuro

Durante la elaboración del proyecto, surgieron varias ideas y alternativas que por una u otra razón no formaron parte de la propuesta final que se plantea en este trabajo. Algunas simplemente se descartaron por razones teóricas (por no ajustarse o ayudar a resolver el problema), otras, en cambio, fueron implementadas, evaluadas, y luego descartadas por su mala performance. Existe un tercer tipo de consideraciones que no fueron implementadas, pero que pueden formar parte en un próximo abordaje del problema. Son éstas las que se describen a continuación.

Mejor pre-procesamiento

Si se dispone de las fechas de lectura de los consumos, una buena medida consiste en tomar el consumo que presento el cliente y dividirlo entre los días transcurridos entre lectura y lectura. De este modo, en lugar de trabajar con el consumo “mensual” podemos trabajar con el *consumo diario promedio*. Los valores de consumo mensual con los que se trabaja en este proyecto, corresponden al consumo registrado entre dos lecturas consecutivas, las cuales pueden ser en un mes un periodo de 30 días y en otro 35 o 28 días. Esta medida elimina el ruido introducido al tomar consumos en intervalos desiguales.

Más características

Además de las características propuestas, se tomaron en cuenta otras características que no fueron implementadas por carecer de la información necesaria para la base de datos que se utilizó. Algunos de los ejemplos propuestos son:

- Tipo de contador (Digital o Analógico)
- Potencia contratada
- Zona de Montevideo
- Correlación con la temperatura promedio mensual
- Antecedentes de fraude

Evaluación del β óptimo

Como se menciono en capítulos anteriores, el parámetro β puede ser utilizado para ponderar $Recall$ frente a $Precision$ (o viceversa) a la hora de calcular el F_{value}. En futuros abordajes al problema se debe estudiar, en conjunto con el personal de UTE, el impacto que tiene variar β. Recordemos que al variar este parámetro regulamos el compromiso entre mejorar el $Recall$ (la cantidad de anómalos detectados) frente al $Precision$ (cantidad de anómalos verdaderos dentro del conjunto etiquetado como anómalo). Mejorar el $Recall$ a expensas de la $Precision$ trae como consecuencia, más anómalos detectados, pero a su vez una mayor número de consumos detectados como anómalos. En particular se debe ajustar el valor de β en función de la capacidad de inspección que se tenga en determinado instante. A partir de la disponibilidad de personal para realizar inspecciones se puede indicar el "tamaño" de la base que se puede abarcar y optimizar este parámetro de modo que la cantidad de consumos detectados como anómalos, y por lo tanto a inspeccionar, quede del tamaño adecuado.

Otras estrategias de combinación

Además de las estrategias de combinación planteadas en el capítulo 11, existen otras alternativas que pueden ser tenidas en cuenta. Sobre el final del proyecto surgieron algunas ideas que no llegaron a ser evaluadas pero que pueden ser incluidas en futuros abordajes al problemas. Por ejemplo, una alternativa puede ser combinar (por cualquiera de los métodos descriptos en el capítulo 11) varias instancias de cada uno de los clasificadores. Una forma es variando el porcentaje de consumos anómalos en el entrenamientos y por lo tanto llegar a clasificadores distintos. En lugar de combinar los 4 clasificadores propuestos, incluir varias instancias de OPF y el Árbol con $p\_bal = \{0\,0{,}4\}$, teniendo de ese modo 6 salidas para combinar ($OPF_0,\ OPF_{0,4},\ C4{,}5_0,\ C4{,}5_{0,4}$ One Class SVM y CS-SVM). Otra alternativa puede ser combinar muchas instancias de cada uno de los clasificadores, variando el conjunto de consumos con los que se realiza el entrenamiento, por ejemplo realizando bagging (ver sección 7.2)

Generalización

En este problema, se analizaron en profundidad las técnicas disponibles para la detección de fraude en energía eléctrica y el tipo de tratamiento que debía recibir un problema con clases desbalanceadas. Sin embargo el entrenamiento de los clasificadores y las decisiones tomadas fueron utilizando una base de datos (Base-2) específica de 1500 consumos aproximadamente. Teniendo en cuenta que en Montevideo hay aproximadamente medio millón de consumidores de energía eléctrica, el próximo paso natural consiste en comenzar a entrenar los algoritmos con bases de mayor tamaño, incluyendo también a los consumos residenciales, dándole una mayor generalidad a la solución. Luego, progresivamente ir realizando inspecciones y en función del resultado que arrojan las mismas, re-entrenar los algoritmos. Este procedimiento, a lo largo del tiempo, desembocará en un sistema mucho más genérico con mayor capacidad para monitorear al enorme número de clientes. Por otro lado, esta retroalimentación, irá independizando el algoritmo del etiquetado realizado a partir del conocimiento actual de los expertos, e introducirá elementos nuevos al tomar en cuenta el resultado de las inspecciones.

Mejora del software

En este proyecto se realizó una interfaz gráfica de usuario que permite al mismo una fácil interacción con todas las funcionalidades desarrolladas. Sin embargo, está programada sobre la plataforma matlab, y por lo tanto solo se puede utilizar dentro de este programa. Es necesario en un futuro independizarse de matlab y adaptar esta interfaz para que se pueda instalar y usar sin ningún otro requerimiento. A su vez, sería importante mejorar el ingreso de nuevas bases de datos, para que la retroalimentación sea más fluida.

División de la base según tipo de consumo

Hasta el momento, se utilizaron distintas bases de datos que contienen consumos de diversos comercios, como pueden ser almacenes, supermercados o autoservicios, pero el entrenamiento y clasificación fue realizado con toda la base sin diferenciar el tipo. Una alternativa que puede dar buenos resultados es separar la base según el tipo de consumidor, por ejemplo, para clasificar a los almacenes entrenar el algoritmo sólo con almacenes. Esto no se pudo poner en práctica en el presente trabajo debido al tamaño reducido de la base de datos, pero si en un futuro se cuenta con datos suficientes puede ser una buena alternativa.

Apendice A1: Interfaz de Usuario

En este apéndice de describe las funcionalidades básicas de la interfaz de usuario realizada en el marco del proyecto para la detección de consumos anómalos. En primer lugar se introducen los pasos requeridos parea la instalación y puesta en marcha de la interfaz. Luego mostramos como evaluar distintos esquemas de combinación. Se puede variar el conjunto de características, utilizar o no algoritmos de selección, elegir distintos clasificadores o distintos esquemas de combinación y observar el desempeño de cada uno de los esquemas. Luego explicaremos cómo entrenar un clasificador (una vez que determinamos cual es el esquema más conveniente utilizando la funcionalidad de evaluar). Y por último se abarca la clasificación de una determinada base de datos que deseamos inspeccionar. También se explicara como ver la información arrojada por los clasificadores y como visualizar los consumos, las características, etc.

15.3. Introducción

Antes introducir funcionalidades disponibles en la interfaz, comencemos por definir las partes y opciones que tenemos en la interfaz y a familiarizarnos con cada una de ellas. En primer lugar, observemos la zona delimitada en la figura 15.2. Comenzando de arriba hacia abajo, la primer pestaña nos permite seleccionar la base de datos que deseamos utilizar tanto para entrenar el algoritmo como para evaluar determinada estrategia de clasificación. Las bases que se muestran como opciones, son todas bases de datos ya etiquetadas. Las opciones disponibles son:

- Base-1
  500 consumos aproximadamente del tipo autoservicios que presentan fuerte estacionalidad.

- Base-2
  1504 consumos, de tipo comercial (panaderías, minimercados, autoservicios, etc).

La segunda pestaña nos permite seleccionar la base de datos que deseamos clasificar. Esta base de datos solo sera utilizada en la modalidad *clasificar*,

no se utiliza durante el entrenamiento ni durante la evaluación de los clasificadores. Esta base puede o no tener etiquetas. Las opciones disponibles en la segunda pestaña son:

- Base-1
  Idem caso anterior.
- Base-2
  Idem caso anterior.
- Base-3
  Base de 3000 consumos aproximadamente, de tipo comercial, sin etiquetas.

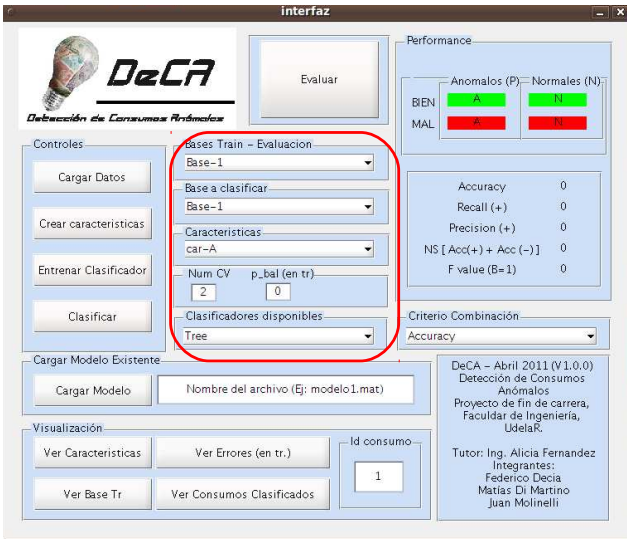


Figura 15.2: Zona de opciones

En tercer lugar tenemos la opción de seleccionar qué características deseamos usar para representar a los consumos. Las opciones disponibles son:

- Car-A
  Conjunto de características propuesto en [10].
- Car-DeCA
  Conjunto de características propuesto en el capitulo 9.
- Car-DeCA-c/selección
  Con esta opción, para cada uno de los clasificadores disponibles se utilizan las características *óptimas* obtenidas en el proceso de selección. Las características que se seleccionan para cada clasificador se pueden encontrar en el capitulo 10.
- Car-C
  Con esta opción se utilizan como características las propias medidas de consumo mensual.

A continuación se encuentran dos casilleros donde podemos ingresar 2 parámetros al sistema. El primero llamado *Num CV*, donde podemos ingresar el número de validaciones cruzadas que deseamos tomar para evaluar el clasificador. Cuidado que este parámetro solo se utiliza en el modo **Evaluación**, en el modo **Entrenamiento** o **Clasificación** se utiliza toda la base para entrenar (o clasificar) y se ignora el parámetro *Num CV*. El valor por defecto es 2, aunque se recomienda evaluar con al menos 5 particiones, si el tiempo no es una limitante se puede utilizar también 10 particiones.

El segundo parámetro que podemos ingresar al sistema es *p bal*. Este parámetro permite balancear las bases a **la hora de entrenar los clasificadores OPF y Tree (el resto de los algoritmos de clasificación ignoran el valor de este parámetro)**. El balanceo

se realiza en la base de entrenamiento (**tanto en el modo Evaluación como en el modo Entrenamiento**). El valor por defecto es 0 (sin balancear). Para obtener mejores F_{value} utilizando los clasificadores OPF o Tree se recomienda utilizar p_bal de $0{,}3$ o $0{,}35$ (se balancea en train con un 30 % de consumos anómalos o 35 % respectivamente). También se puede utilizar valores como $0{,}5$ si se quiere aumentar el número de anómalos detectados, pero se debe tener en cuenta que esto incrementa el numero de falsos anómalos (consumos normales clasificados como anómalos). Empíricamente hemos notado que valores del orden de $0{,}35$ representan un buen compromiso entre TP y FP.

Llegando a la última opción disponible de las enmarcadas en la figura 15.2, encontramos los distintos clasificadores que podemos utilizar. Una vez seleccionado un método de clasificación, éste es utilizado en el *Evaluacion* (si se desea evaluar dicho clasificador), en el modo *Entrenamiento* (se entrenara el clasificador seleccionado con la base escogida como train) y en el modo *Clasificación* (se utiliza el clasificador entrenado previamente para clasificar la base seleccionada). Las opciones disponibles son:

- Tree: clasificador C4.5 con Adaboost
- one class SVM: clasificador One Class SVM
- OPF_sl: clasificador OPF
- C-SVM: clasificador C-SVM con pesos (CS-SVM)
- Combinación: Utiliza **todos** los clasificadores anteriores y los combina mediante la regla de combinación seleccionada (en el siguiente párrafo abordaremos las distintas opciones de combinación disponibles y cómo seleccionar cada una de ellas).

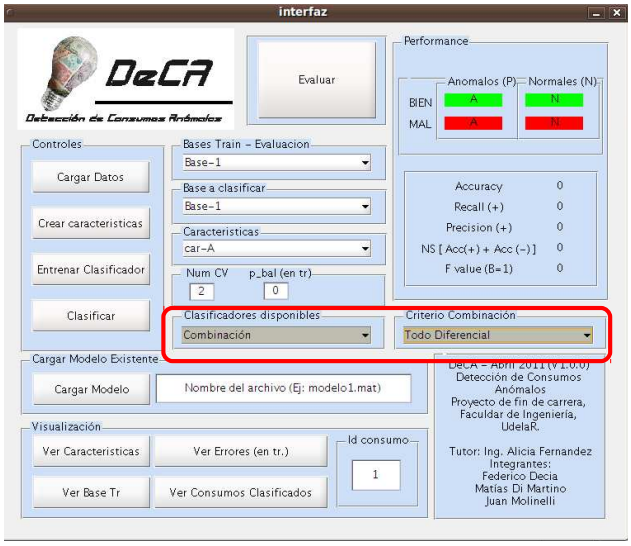


Figura 15.3: Clasificadores Disponibles y Criterios de Combinación

El fundamento teórico de cada uno de los clasificadores se puede encontrar en el capítulo 6, los detalles de las implementaciones realizadas en este proyecto están presentadas en el capítulo 11.

Si seleccionamos como clasificador *Combinación*, el sistema utilizará todos los clasificadores y los combinará mediante la regla de combinación seleccionada.

En la figura 15.3 se pueden observar las pestañas que permiten seleccionar el clasificador y el modo de combinación llamadas *Clasificadores Disponibles* y *Criterio Combinación* respectivamente. Si se elige algún clasificador distinto de *Combinación*, el sistema simplemente ignora lo que se seleccione en *Criterio Combinación*.

15.4. Clasificar Consumos

Para realizar la clasificación de una base de datos, primero debemos o bien seguir los pasos enumerados en la sección 15.6 (Entrenar un clasificador) o bien cargar un modelo generado previamente. Para cargar un modelo, simplemente ingresamos el nombre del modelo previamente generado (por ejemplo `modelo-1.mat`) que debe estar guardado en `DeCA\v.1.0.0\`, luego presionamos *Cargar Modelo*.

La herramienta viene provista de algunos modelos que se listan a continuación. El usuario puede utilizar cualquiera de éstos si desea clasificar consumos de manera rápida sin preocuparse por la evaluación y generación de modelos específicos.

- `modelo-TD-04.mat`
  modelo generado utilizando la combinación “Todo Diferencial” con b_bal=0.4 entrenado con la Base-2.
- `modelo-TD-00.mat`
  modelo generado utilizando la combinación “Todo Diferencial” con b_bal=0 entrenado con la Base-2.
- `modelo-I-04.mat`
  modelo generado utilizando la combinación “Iterativo” con b_bal=0.4 entrenado con la Base-2.
- `modelo-I-00.mat`
  modelo generado utilizando la combinación “Iterativo” con b_bal=0 entrenado con la Base-2.

Los pasos para clasificar una base de datos son los siguientes:

- **Paso 1:**
  Seleccionar la base de datos que se deseamos clasificar.
- **Paso 2:**
  Seleccionar las características que se desea utilizar.
- **Paso 3:**
  Cargar la base de datos (presionando *Cargar Datos*)
- **Paso 4:**
  Crear las características (presionando *Crear Características*)
- **Paso 5:**
  Clasificar presionando (*Clasificar*)

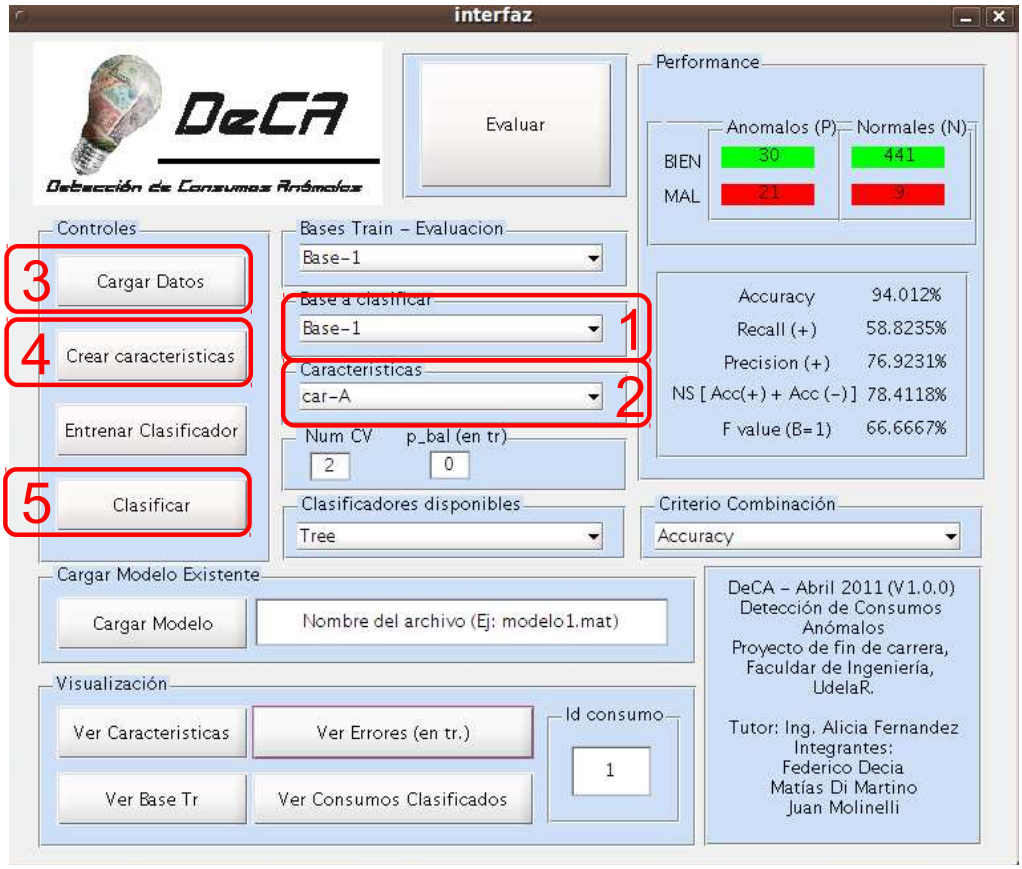


Figura 15.4: Pasos para clasificar

Los pasos se ilustran en la figura 15.4.

Una vez que se clasifico a la base de datos seleccionada, se pueden visualizar los consumos clasificados utilizando la funcionalidad *Ver Consumos Clasificados* como se describe a continuación y se ilustra en la figura 15.5.

Luego de presionar el boton *Ver Consumos Clasificados* (enmarcada con el número 1 en la figura 15.5) se despliegan el primer[a] consumo de la base seleccionada en la opción *Base a clasificar*. Al mismo tiempo en el promt de matlab se despliega la lista con el nombre de cada una de las características (enmarcado con el número 2 en la figura) y se solicita ingresar (ver zona enmarcada con el número 3): 1- si deseamos ver el consumo anterior 2- si deseamos ver el siguiente y 3- para salir de la visualización, también podemos ingresar un número n ($n \neq 1,\ 2,\ 3$) para saltar hasta el consumo n. Además de ver la curva de consumo, que se muestra en color rojo si el consumo es anómalo o en color verde si es normal y en azul si no se tiene etiquetas previas para el consumo, se muestra mediante barras (ver zona enmarcada con el número 4 en la figura) el valor de cada una de las características para dicho consumo. En la zona enmarcada con el número 5 se muestra la información arrojada por el clasificados, que indica si el consumo que se esta observando fue clasificado como anómalo o normal. También se despliega información adicional del consumo (en caso de que este disponible) como se muestra por ejemplo en la zona 6 donde

[a] si en lugar de comenzar visualizando el primer consumo deseamos arrancar en el número n, en el casillero llamado *id consumo* ingresamos n y de ese modo la visualización comienza desde el consumo n

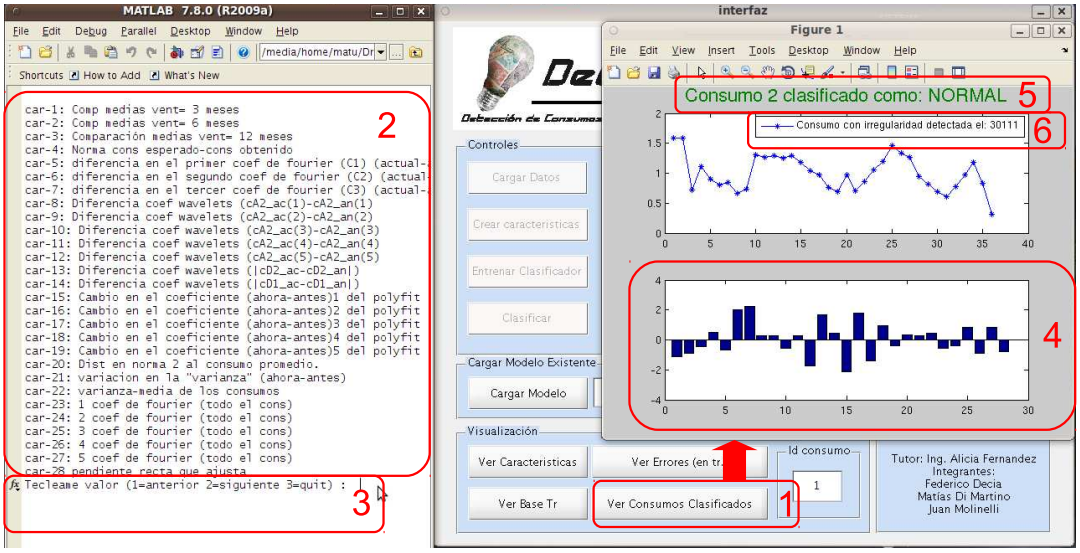


Figura 15.5: Visualización de los consumos clasificados

indicamos que el consumo tiene una detección de irregularidad en la fecha mencionada.

15.4.1. FAQ

- **¿Cómo agrego nuevas bases de datos?**
  Las bases de datos deben ser incorporadas a `interfaz.m` antes de poder ser utilizadas.
  Otra alternativa más rápida, es renombrar la base que queremos utilizar como "cons_etiq_pru2.csv"luego utilizar en la interfaz de usuario la Base-1 (que es la que hace referencia al archivo antes mencionado). Se debe tener en cuenta que para que el sistema funcione correctamente los datos deben presentar el mismo formato que "cons_etiq_pru2.csv"

- **¿Cómo genero nuevos modelos?**
  Los pasos par generar nuevos modelos se muestran en la sección 15.6. Una vez que se entreno un clasificador se debe renombrar el archivo `modelo.mat` al nombre que nos parezca adecuado, luego podremos cargar dicho modelo haciendo referencia al nombre escogido cuando deseemos utilizarlo para clasificar consumos.

- **¿Cuánto demora en correr el programa?**
  La clasificación de consumos culmina en unos minutos. Otras funcionalidades como *Evaluar Clasificador* o *Entrenar Clasidicador* pueden requerir desde un par de minutos hasta un día, dependiendo de las opciones escogidas.

- **¿Puedo plantear nuevas características o clasificadores?**
  Sí, pero, para hacerlo se debe modificar `interfaz.m`, que es donde se invocan los distintos algoritmos. Las características están definidas en `caracteristicas_v4.m` de modo que para agregar nuevas características debe programarse en el código antes mencionado.

- **¿Se dispone de información adicional además de las etiquetas que el clasificador asigna a cada muestra?**
  Sí. Una vez que finalizo de correr la clasificación o la evaluación de un clasificador, en `v.1.0.0\` se encuentra `Salida.mat` que tiene información adicional. Para acceder a ésta, simplemente cargar las variables utilizando `load Salida.mat` en una consola de matlab. Luego en el *workspace* encontrara variables que indican las etiquetas, la confianza sobre cada una de las etiquetas, las salidas de cada uno de los clasificadores individuales y el índice de cada uno de los consumos.

- **¿Donde puedo obtener más información acerca de este programa?**
  Enviar un correo a deca.proyecto@gmail.com

15.5. Modo Evaluación

En esta sección describiremos como preceder para evaluar determinada configuración. Por *determinada configuración* entendemos: se escogió una base de datos etiquetada con la que se desea evaluar, se opto por un determinado conjunto de características, se fijo los valores de *Num CV* y p_bal deseados y se definió un algoritmo de clasificación (en caso de haber optado por la combinación adicionalmente se debe definir el criterio de combinación que se desea utilizar). Una vez que se escogieron todas las propiedades que definen un esquema de clasificación podemos proceder a evaluar la misma como se explicara a continuación.

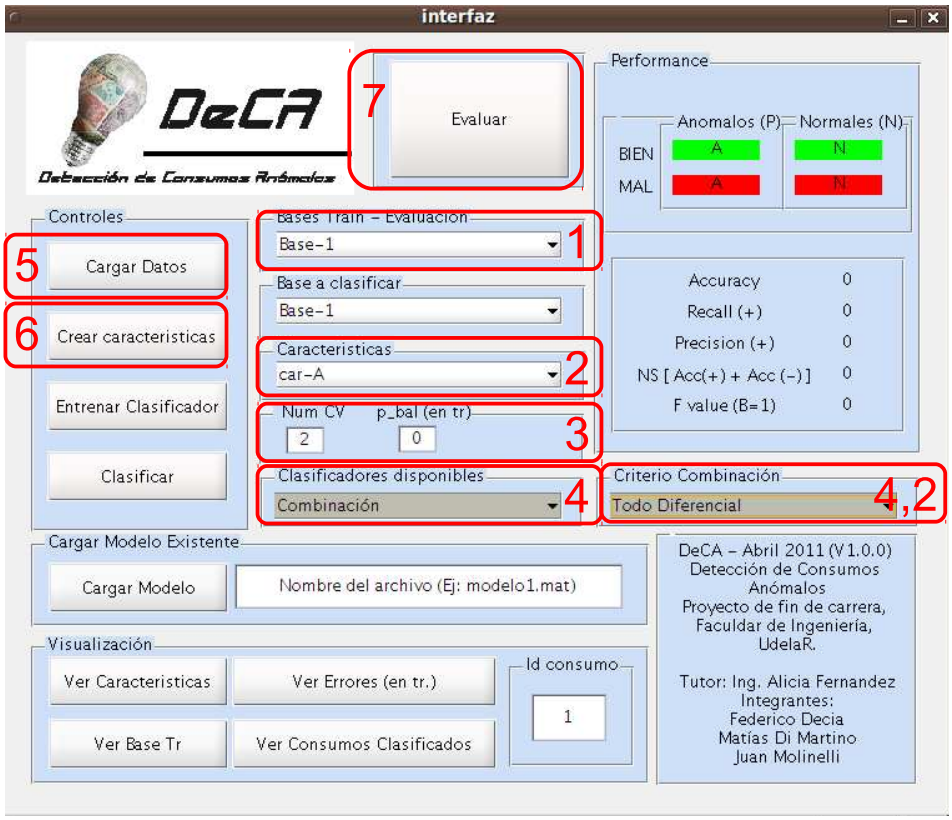


Figura 15.6: Pasos para evaluar una configuración

- **Paso 1:**
  Seleccionar la base de datos que se desea utilizar para la evaluación.

- **Paso 2:**
  Seleccionar las características que se desea utilizar.

- **Paso 3:**
  Fijar el número de validaciones cruzadas que se desea realizar y escoger p_bal en caso de utilizar los clasificadores OPF, Tree o Combinación (que utiliza ambos).

- **Paso 4:**
  Seleccionar el algoritmo de clasificación que se desea utilizar.

  - **Paso 4.2:**
    Si se escogió como clasificador *Combinación*, seleccionar el criterio de combinación que se desea utilizar.

- **Paso 5:**
  Cargar la base de datos (presionando *Cargar Datos*)

- **Paso 6:**
  Crear las características (presionando *Crear Características*)

- **Paso 7:**
  Evaluar el clasificador presionando (*Evaluar*)

Los pasos se ilustran en la figura 15.6

El proceso de evaluación puede demorar desde minutos a días dependiendo de los algoritmos seleccionados. A continuación presentamos una tabla con un estimativo de los tiempos aproximados que toman algunos algoritmos para que se tome como referencia y se tenga en cuenta a la hora de realizar corridas. Mientras se esta realizando la evaluación,

Base	Características	Clasificador	Orden de tiempo
Base-2	DeCA-c/selección	OPF	1 min
Base-2	DeCA-c/selección	Tree	2 min
Base-2	DeCA-c/selección	One Clas SVM	1 hora
Base-2	DeCA-c/selección	C-SVM	1 día

el cursor del mouse queda en el modo de espera y los botones permanecen con un tono gris desactivados. Una vez que finaliza el proceso los botones vuelven a su formato inicial y se despliegan los resultados.

Una ves que finalizo el proceso de evaluación, tenemos información que nos permite evaluar la performance del esquema ensayado para la base de datos que se esta utilizando. Como se muestra en la figura 15.7 la información que se despliega consiste en:

- **Matriz de confusión**
  La primer tabla que se recuadra en la figura 15.7 despliega la cantidad de consumos anómalos correctamente clasificados, la cantidad de normales bien clasificados

(ambos resaltados con color verde), la cantidad de consumos anómalos mal clasificados y la cantidad de consumos normales mal clasificados (ambos en color rojo).

- **Indicadores de performance:** El secundo recuadro que se resalta en la figura 15.7 muestra el valor de los indicadores de performance más utilizados en la bibliografía y las publicaciones en el área de reconocimiento de patrones. En la sección 4.3 se introducen las definiciones y los fundamentes que justifican la observación de estas cantidades en particular. Los indicadores desplegados son:

  - Accuracy
    Porcentaje de consumos bien clasificados
  - Recall (+)
    Porcentaje de consumos anómalos bien clasificados
  - Precision (+)
    Del total de consumos **clasificados** como anómalos, que porcentaje **es** anómalo.
  - NS
    Porcentaje de consumos anómalos bien clasificados más porcentaje de normales bien clasificados sobre dos.
  - Fvalue
    Se calcula en función del Recall y la Precision como (para $\beta = 1$):

$$Fvalue = \frac{Recall \times Precision}{Recall + Precision}$$

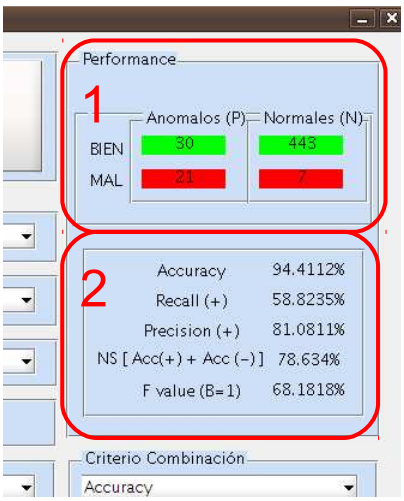


Figura 15.7: Evaluación de la performance

Además de obtener información referida al desempeño global del clasificador, podemos visualizar la distribución de las características para cada una de las clases (clase *anómalo* y *normal*). También se puede observar cada consumo con sus respectivas características y observas aquellos consumos mal clasificados como se explica a continuación.

Visualización de la distribución de las características:

Pulsando el boton *Ver Características* como se muestra en la figura 15.8 se despliegan los histogramas de la cantidad de consumos que toman cada uno de los valores para cada característica, en azul se representan a los consumos de la clase normal y en rojo a los anómalos. La idea de esta herramienta es poder visualizar a simple vista que características *discriminan* más a los consumos en función de su clase.

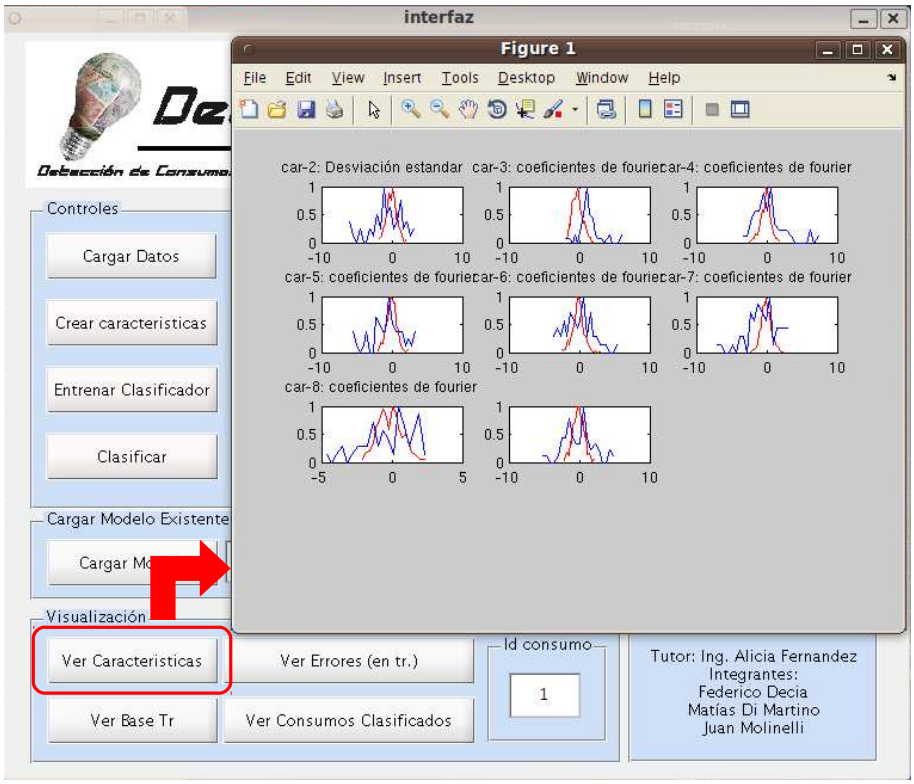


Figura 15.8: Visualización de las características

Visualización de la base *train*

También podemos visualizar los consumos de la base de datos seleccionada en la opción *Bases Train - Evaluación* presionando el boton *Ver Base Tr* como se muestra en la figura 15.9. Luego de presionar el boton *Ver Base Tr* (enmarcada con el número 1 en la

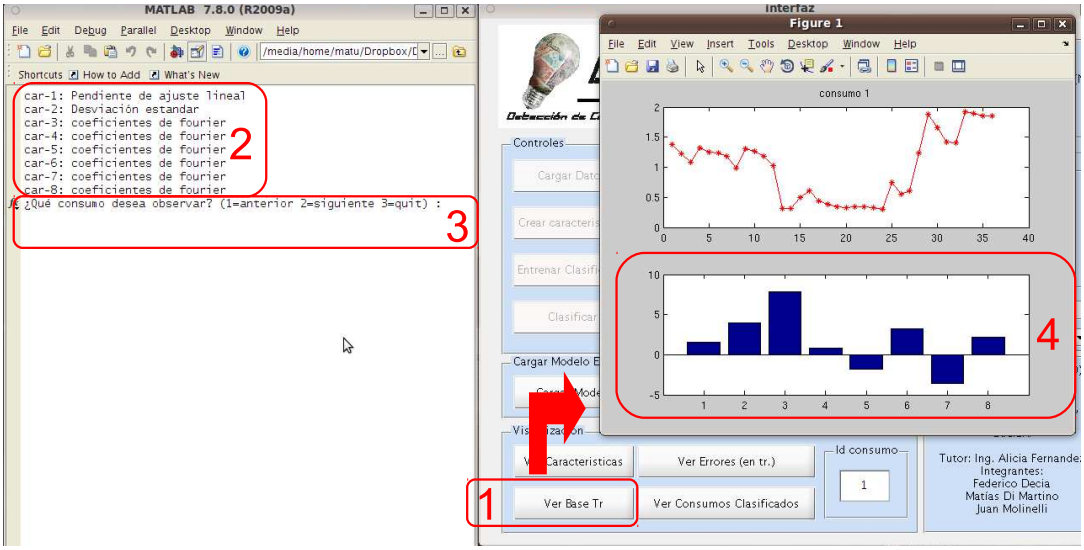


Figura 15.9: Visualización de la base que se esta evaluando (o utilizando para entrenar)

figura 15.9) se despliegan el primer[b] consumo de la base seleccionada en la opción *Bases Train - Evaluación*. Al mismo tiempo en el promt de matlab se despliega la lista con el nombre de cada una de las características (enmarcado con el número 2 en la figura) y se solicita ingresar (ver zona enmarcada con el número 3): 1- si deseamos ver el consumo anterior 2- si deseamos ver el siguiente y 3- para salir de la visualización, también podemos ingresar un número n ($n \neq 1,\ 2,\ 3$) para saltar hasta el consumo n. Además de ver la curva de consumo, que se muestra en color rojo si el consumo es anómalo o en color verde si es normal, se muestra mediante barras (ver zona enmarcada con el número 4 en la figura) el valor de cada una de las características para dicho consumo.

[b] si en lugar de comenzar visualizando el primer consumo deseamos arrancar en el número n, en el casillero llamado *id consumo* ingresamos n y de ese modo la visualización comienza desde el consumo n

Visualización de consumos mal clasificados.

Otra opción disponible luego de evaluar un clasificador, es observar aquellos consumos que fueron mal clasificados (y el valor de sus respectivas características). Presionando sobre *Ver errores (en tr.)* como se muestra en el recuadro número 1 en la figura 15.10 se despliegan los consumos que fueron mal clasificados.
En el promt de matlab se despliega la lista con el nombre de cada una de las características

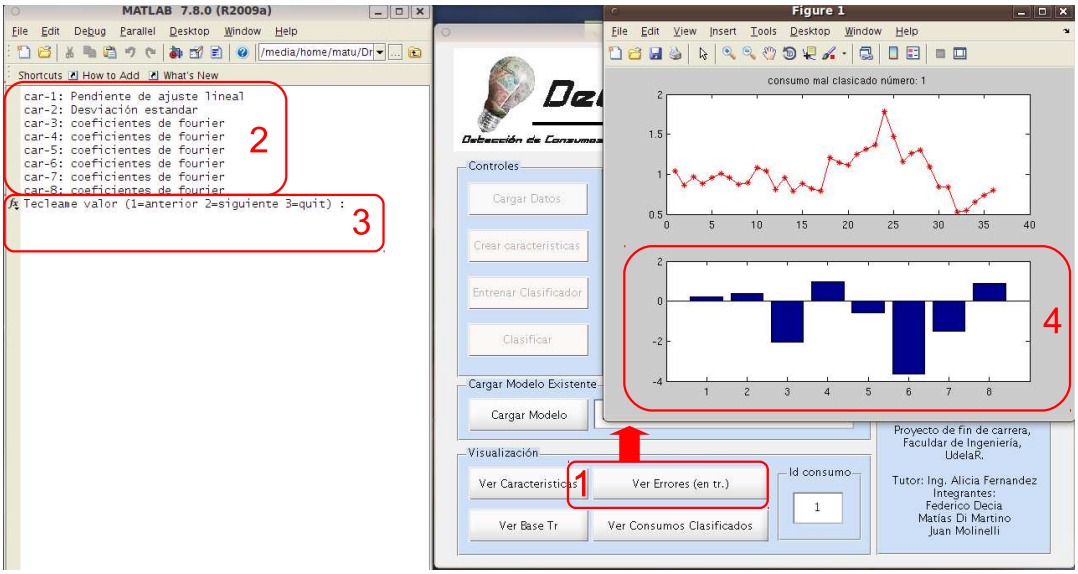


Figura 15.10: Visualización de los consumos mal clasificados durante la evaluación (o train)

(enmarcado con el número 2 en la figura) y se solicita ingresar (ver zona enmarcada con el número 3): 1- si deseamos ver el consumo anterior (mal clasificado) 2- si deseamos ver el siguiente consumo mal clasificado y 3- para salir de la visualización. Además de ver la curva de consumo, que se muestra en color rojo si el consumo es anómalo o en color verde si es normal, se muestra mediante barras (ver zona enmarcada con el número 4 en la figura) el valor de cada una de las características para dicho consumo. Tener en cuenta que un consumo mal clasificado de color rojo, significa que el consumo es anómalo y fue mal clasificado, es decir, dicho consumo fue clasificado (en contradicción con su etiqueta) como normal. Por otro lado si el consumo se dibuja en verde, es un consumo con etiqueta *normal* que fue clasificado como anómalo.

15.6. Modo de Entrenamiento

Una vez que se ensayaron distintas alternativas, combinando los conjuntos de características, algoritmos de clasificación, parámetros de los clasificadores y las estrategias de combinación en el modo de *Evaluación* descripto en la sección anterior, podemos comparar el desempeño de cada una de las estrategias evaluadas. Luego, es natural utilizar aquella estrategia que parece más prometedora para entrenar el algoritmo y clasificar una base de datos no etiquetada.

Antes de describir los pasos necesarios para entrenar el clasificador, puntualicemos un aspecto importante. En el modo de evaluación, se utiliza la base etiquetada para evaluar valga la redundancia a un modo de clasificación, para esto se parte la base de datos en *Num CV* subconjuntos, y para clasificar a cada uno de los subconjuntos se entrena el algoritmo con todos los subconjuntos restantes. De este modo se clasifica toda la base de entrenamiento y se comparan las etiquetas de la base con las proporcionadas por el clasificador. En el modo de entrenamiento en cambio se utiliza **toda** la base de entrenamiento para entrenar los algoritmos y no se realizan ensayos de performance. De este modo los algoritmos quedan preparados para clasificar una base independiente que puede o no tener equiquetas como se explicara en la sección siguiente.

Los pasos para entrenar el clasificador son los siguientes:

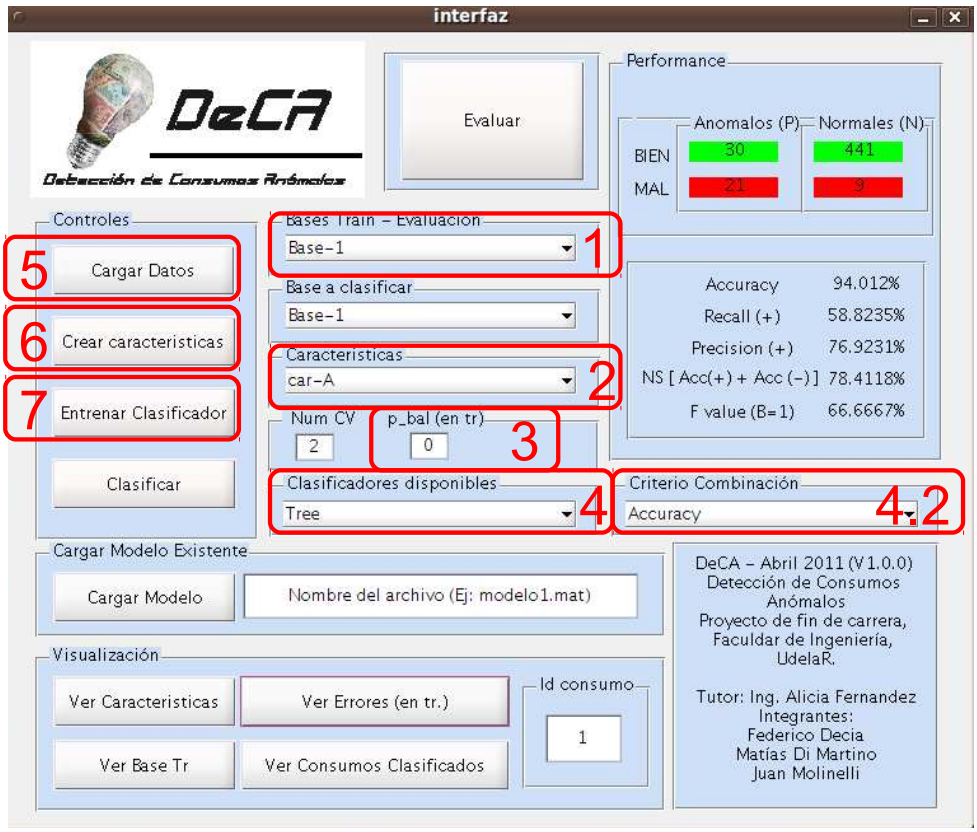


Figura 15.11: Pasos para entrenar

- **Paso 1:**
  Seleccionar la base de datos que se desea utilizar para el entrenamiento.

- **Paso 2:**
  Seleccionar las características que se desea utilizar.

- **Paso 3:**
  Escoger p_bal en caso de utilizar los clasificadores OPF, Tree o Combinación (que utiliza ambos).

- **Paso 4:**
  Seleccionar el algoritmo de clasificación que se desea utilizar.

  - **Paso 4.2:**
    Si se escogió como clasificador *Combinación*, seleccionar el criterio de combinación que se desea utilizar.

- **Paso 5:**
  Cargar la base de datos (presionando *Cargar Datos*)

- **Paso 6:**
  Crear las características (presionando *Crear Características*)

- **Paso 7:**
  Entrenar el clasificador presionando (*Entrenar Clasificador*)

Los pasos se ilustran en la figura 15.11.

Luego de finalizado el entrenamiento, se puede pasar al modo *clasificación* descripto en la próxima sección. Como algunos de los procesos de entrenamiento demandan cantidades importantes de tiempo, cuando finaliza el entrenamiento, en la carpeta en `\DeCA\v.1.0.0\` se crea el archivo `modelo.mat` que puede ser guardado y utilizado para posteriores clasificaciones sin la necesidad de entrenar nuevamente.

Nota: cuidado que cada vez que se entrena un clasificador, el archivo `DeCA\v.1.0.0\modelo.mat` *se sobre escribe. Si deseamos guardar un modelo es conveniente renombrarlo y luego cargarlo haciendo referencia al nuevo nombre.*

Bibliografía

[1] V. Garcia J.S. Sanchez R.A. Mollineda R. Alejo and J.M. Sotoca. The class imbalance problem in pattern classification and learning. 2007.

[2] Pratti RC Batista GE and Monard MC. A study of the behavior of several methods for balancing machine learning training data. *SIGKDD Explorations 6*, pages 20–29, 2004.

[3] Nitesh V. Chawla and Jared Sylvester. Eploiting diversity in ensembles: Improving the performance on unbalanced datasets. *Departament of Computer Science and Engineering*, 2007.

[4] Bowyer KW Chawla NV and Hall LO. Smote: synthetic minority over-sampling technique. *Journal of Artificial Intelligence Research*, 2002.

[5] Lazarevic A Chawla NV and Hall LO. Smoteboost: impoving prediction of the minority class in boosting. *European Conf. ok Principles and Practice of Knowledge Discovery in Databases*, 2003.

[6] Prabhakar Raghavan Christopher D. Manning and Hinrich Schutze. *An Introduction to Information Retrival*. Cambridge University Press, Cambridge, England, 1 edition, 2009.

[7] N. Cristianini and J. Shawe-Taylor. *An Introduction to Support Vector Machines*. Cambridge, MA: Cambridge Univ, 2000.

[8] B. V. Dasarathy and B. V. Sheela. A composite classifier system design: concepts and methodology. *Proceedings of IEEE*, pages 67:708–713, 1978.

[9] Caio César Oba Ramos Andrá Nunes de Sousa Joao Paulo Papa and Alexandre Xavier Falcao. A new approach for nontechnical losses detection based on optimum-path forest. *IEEE TRANSACTIONS ON POWER SYSTEMS*, 2010.

[10] Juan Pablo Kosut Diego Alcetegaray. One class svm para la detección de fraudes en el uso de energía eléctrica. *Trabajo Final Curso de Reconocimiento de Patrones, Dictado por el IIE- Facultad de Ingeniería- UdelaR*, 2008.

[11] T. G. Dietterich. Ensemble methods in machine learning. *Multiple Classifier Systems, volume 1857 of Lecture Notes in Computer Science*, 2000.

[12] Richard O. Duda, Peter E. Hart, and David G. Stork. *Pattern Classification*. Wiley, New York, 2. edition, 2001.

[13] R.O. Duda and P.E. Hart. *Pattern Classification and Scene Analysis*. New York, 1973.

[14] S. Dumais. Using svms for text categorization. *IEEE Intell. Syst. Mag., Support Vector Machines, vol. 13, no. 4, pp. 21-23,* 1998.

[15] Y. Freund and R. E. Schapire. A decision theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences*, pages 119–139, 1997.

[16] K.S. Fu. Syntactic pattern recognition and applications. *Englewood Cliffs Prentice-Hall*, 1982.

[17] Xinjian Guo and Guangtong Zhou. On the class imbalance problem. *IIE - Computer Society*, 1:192, 2008.

[18] G.V.Trunk. A problem of dimensionality: A simple example. *IEEE Trans. Pattern Analysis and Machine Intelligence, Vol. 1, no. 3*, 1979.

[19] L. Devroye L. Gyorfi and G. Lugosi. A probabilistic theory of pattern recognition. *Springer-Verlag*, 1996.

[20] Anil K. Jain and Robert P.W. Duin. Statistical pattern recognition: A review. *IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol 22, No. 1*, 2000.

[21] L. Xu A. Krzyzak and C. Y. Suen. Methods of combining multiple classifiers and their application to handwriting recognition. *IEEE Transactions on Systems Man and Cybernetics*, 22:418, 1992.

[22] Matwin S Kubat M, Holte R. Detection of oil-spills in radar images of sea surface. *Machine Learning 30*, pages 195–215, 1998.

[23] Ludmila I. Kuncheva. *Combining Pattern Classifiers: Methods and Algorithms*. Wiley-Interscience, 2004.

[24] Kubat M and Matwin S. Adressing the curse of imbalanced training sets: onesided selection. *Proc. 14th Intl. Conf. on Machine Learning*, pages 179–186, 1997.

[25] S. Mallat. *A theory for multiresolution signal decomposition: the wavelet representation.* IEEE Pattern Anal. and Machine Intell., vol. 11, no. 7, pp. 674-693, 1989.

[26] Stephane Mallat. *A Wavelet Tour of Signal Processing, Second Edition (Wavelet Analysis and Its Applications).* Academic Press, 1999.

[27] Hamed Masnadi-Shirazi and Nuno Vasconcelos. Asymmetric boosting. 2007.

[28] John Mingers. An empirical comparison of prunning methods for decision tree induction. *Machine Learning 4*, pages 225–243, 1989.

[29] C. Muniz, M. Vellasco, R. Tanscheit, and K. Figueiredo. Ifsa-eusflat 2009 a neuro-fuzzy system for fraud detection in electricity distribution, 2009.

[30] Jawad Nagi and Malik Mohamad. Nontechnical loss detection for metered customers in power utility using support vector machines. *IEEE TRANSACTIONS ON POWER DELIVERY, VOL. 25, NO. 2*, 2010.

[31] S. J. Nowlan and G. E. Hinton. Evaluation of adaptive mixtures of competing experts. *Advances in Neural Information Processing Systems 3*, pages 774–780, 1991.

[32] E. Osuna. Applying svms to face detection. *IEEE Intell. Syst. Mag., Support Vector Machines, vol. 13, no. 4*, pages 23–26, 1998.

[33] T. Pavlidis. Structural pattern recognition. *Springer-Verlag*, 1977.

[34] Barandela R and Garcia V. Strategies for learning in class imbalance problems. *Pattern Recognition*, pages 849–851, 2003.

[35] B. Raskutti and A. Kowalczyk. Extreme rebalancing for svms: a case study. *SIGKDD Explorations, 6(1), 2004*, pages 60–69, 1998.

[36] Harry Tagaris Rong Jiang and Andrei Laschusz. Wavelets based feature extraction and multiple cassifiers for electricity fraud detection, 2000.

[37] Ana María Clara Ruedin. Introducción a las wavelets, 2004.

[38] Bernhard Scholkopf and Alexander J. Smola. *Learning with Kernels*. The MIT Press, London, 2. edition, 2002.

[39] Ingo Steinwart and Andreas Christmann. *Support Vector Machines*. Springer, USA, 1. edition, 2006.

[40] Joao P. Papa Alexandre X. Falcao Paulo A.V. Miranda Celso T.N. Suzuki and Nelson D.A. Mascarenhas. Design of robust pattern classifiers based on optimum-path forests. *8th International Symposium on Mathematical Morphology Rio de Janeiro Brazil Oct*, pages 337–348, 2007.

[41] V. N. Vapnik. *The Nature of Statistic Learning Theory*. New York: Springer, 1995.

[42] V. N. Vapnik. *Statistical Learning Theory*. New York: Wiley, 1998.

[43] Shuo Wang and Xin Yao. Theoretical study of the relationship between diversity and single-class measures for class imbalance learning. *IEEE International Conference on Data Mining Workshops*, 2009.

[44] S. Watanabe. Pattern recognition: Human and mechanical. 1985.

[45] Andrew Webb. *Statistical Pattern Recognition*. Wiley, New York, 2. edition, 1999.

[46] Ian H. Witten and Eibe Frank. *Data Mining: Practical Machine Learning Tools and Techniques*. Morgan Kaufmann Series in Data Management Systems. Morgan Kaufmann, second edition, June 2005.

[47] G. Wu and E. Chang. Class-boundary alignment for imbalanced dataset learning. 2003.

[48] Lui Y ands Jin R Yan R. On predicting rare classes with svm ensembles in scene classification. *IEEE Int. Conf. on Acoustic, Speech and Signal Processing*, 1:21–24, 2003.