Reconocimiento de Patrones: Reporte del trabajo final

Pablo Zinemanas

5 de diciembre de 2016

1. Introducción

1.1. Objetivo

El objetivo de este trabajo es probar diferentes técnicas de reconocimiento de patrones a la detección de fuentes sonoras en grabaciones del entorno sonoro urbano.

1.2. Descripción del problema

En los últimos años han surgido algunas iniciativas que buscan desarrollar tecnologías para el monitoreo y diagnóstico del entorno sonoro urbano, orientadas a facilitar la planificación y la gestión de la ciudad [1, 2]. Se basan en una red de sensores distribuidos que permiten registrar los niveles de ruido en tiempo real. Además, mediante el uso de tecnologías de procesamiento de audio y aprendizaje automático se busca generar de forma automática una descripción del tipo de ambiente sonoro registrado, incluyendo las fuentes que lo componen.

La realización del primer *Workshop* de detección y clasificación de escenas acústicas y eventos sonoros (DCASE) en 2016, es una evidencia del interés que ha despertado el tema en la comunidad científica. En el marco de esta reunión académica se organizó el primer desafío DCASE 2016 [3], que involucra varios problemas de análisis automático de entorno sonoro, como clasificación de ambientes y detección de eventos. Se busca que esta iniciativa permita comparar diferentes enfoques utilizando una misma base de datos de grabaciones y medidas de desempeño establecidas.

Uno de los desafíos del DCASE 2016 fue el desarrollo de sistemas para la detección de eventos sonoros en grabaciones del entorno sonoro urbano y doméstico. En la Figura 1 se puede ver un diagrama del sistema planteado. Como se ve en el diagrama, pueden haber eventos sonoros simultáneos. En el caso de las grabaciones del entorno sonoro urbano, los eventos pueden pertenecer a una de la siguientes clases:

- Object banging (Golpes de objetos)
- Bird singing (Pájaros cantando)
- Car passing by (Autos pasando)
- Children shouting (Niños gritando)
- People speaking (Personas hablando)
- People walking (Personas caminando)
- Wind blowing (Viento soplando)

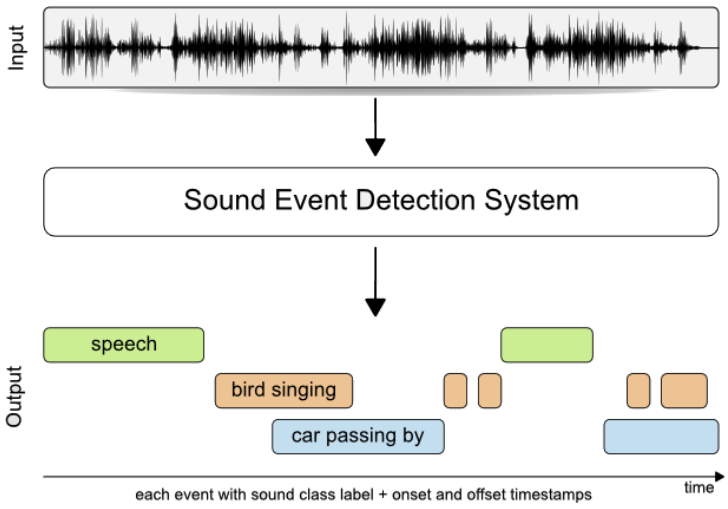


Figura 1: Diagrama de un sistema de detección de eventos

1.3. Antecedentes

Para abordar el problema de detección de fuentes sonoras se puede aprovechar el conocimiento existente en otras áreas relacionadas, tales como el procesamiento de voz [4] y el reconocimiento de instrumentos musicales [5, 6]. Desde fines de la década de 1960 la comunidad científica ha dedicado grandes esfuerzos en diseñar técnicas adecuadas para procesar digitalmente señales de voz, debido a sus diversas aplicaciones en telecomunicaciones e interacción humano-máquina, entre otras. Muchas de las técnicas y características usadas para segmentar señales de voz, reconocer fonemas o identificar hablantes [7], encuentran aplicación en problemas donde se busca caracterizar y detectar otro tipo de sonidos. Por ejemplo, los Coeficientes Cepstrales de Frecuencia MEL (MFCC) o los Coeficientes de Predicción Lineal (LPC), originalmente concebidos para describir el contenido espectral de señales de voz, han sido utilizados con éxito en diversos problemas vinculados a la música, como identificación de cantante [8] o reconocimiento de instrumentos musicales [6].

En el Desafío DCASE 2016 fueron presentados 12 trabajos. Muchos de ellos utilizan (MFCC) como características [9, 10, 11, 3] y diferentes técnicas de reconocimiento de patrones para la clasificación, como SVM, GMM o Random Forest. El uso de redes neuronales para la clasificación también ha sido utilizado [12, 13, 14, 15] y en particular técnicas como *deep learning* también fueron aplicadas [16, 17], debido a que han logrado mejorar el estado del arte en diferentes problemas de procesamiento de señales.

2. Datos

Para desarrollar los algoritmos necesarios y evaluar los resultados se dispone de la base de datos TUT que cuenta con 17 grabaciones, totalizando una hora de duración, y con etiquetas que indican no sólo los sonidos presentes sino también el momento de ocurrencia. Esta base fue dividida de tal forma que 12 grabaciones sean utilizadas para el desarrollo y las 5 restantes sirvan para la evaluación posterior. Además, los datos para desarrollo están divididos en 4 *folds* para medir el desempeño con validación cruzada. Las datos fueron capturados con una grabadora Roland Edirol R09 a una frecuencia de muestreo de 44.1 kHz y una resolución de 24 bits [18]. Uno de las desafíos que presenta la base de datos, es el desbalance de las clases, como puede verse en la Tabla 1.

Clase	Número de instancias
Object banging	23
Bird singing	271
Car passing by	108
Children shouting	31
People speaking	52
People walking	44
Wind blowing	30

Tabla 1: Número de instancias por clase

Para desarrollar y medir el desempeño del sistema los organizadores del desafío proveen una plataforma de trabajo desarrollada en Python y Matlab.

3. Medidas de desempeño

Para estimar el desempeño del sistema, los autores de la base de datos proponen usar medidas de error y F-score en una grilla de tiempo fija. Lo eventos de sonido detectados se comparan con el *ground-truth* en segmentos de un segundo. De esta forma, para cada segmento se pueden considerar tres casos:

- Correcto: un evento es considerado correctamente detectado si está presente en el *ground-truth* y activo en la salida del sistema en dicho segmento.
- Falso Positivo (FP): si el sistema marca un evento como positivo y el *ground-truth* no lo hace.
- Falso Positivo (FN): si el *ground-truth* marca un evento como positivo y la salida del sistema como negativo.

Basados en las medidas de FP y FN se pueden calcular *precision*, *recall* y *F-score* como:

$$P = \frac{TP}{TP + FP},\ R = \frac{TP}{TP + FN},\ F1 = \frac{2PR}{P + R}.$$

Las medidas de error se calculan en términos de inserciones (I), borrados (D[1]) y sustituciones (S). Para un segmento k, su número está dado por:

$$S(k) = \text{mín}\{FN(k), FP(k)\}$$

$$D(k) = \text{máx}\{0, FN(k) - FP(k)\}$$

$$I(k) = \text{máx}\{0, FP(k) - FN(k)\}.$$

Básicamente, una sustitución está definida como el caso en que el sistema detecta un evento en un segmento, pero con la etiqueta equivocada. Esto es equivalente a contener un FP y FN en el mismo segmento. Después de contar las sustituciones por segmento, el resto de FP en la salida del

[1]Del inglés *deletions*.

sistema son contados como inserciones y el resto de FN como borrados. Luego, la medida de error se calcula integrando las medidas anteriores en todos los K segmentos:

$$ER = \frac{\sum_{k=1}^{K} S(k) + \sum_{k=1}^{K} D(k) + \sum_{k=1}^{K} I(k)}{\sum_{k=1}^{K} N(k)},$$

donde $N(k)$ es el número de clases activas en el *ground-truth* en el segmento k [18, 19].

4. Características

La plataforma de trabajo dispone de un sistema de extracción de características, normalización, entrenamiento, clasificación y evaluación. Como características, se calcula la energía en 40 bandas MEL y de ellos 20 coeficientes MFCC. El banco de filtros MEL es representado en la Figura 2, donde se puede ver que son filtros triangulares. El ancho de banda de los filtros es constante hasta 1000 Hz y constante en la escala MEL[2] para frecuencias superiores.

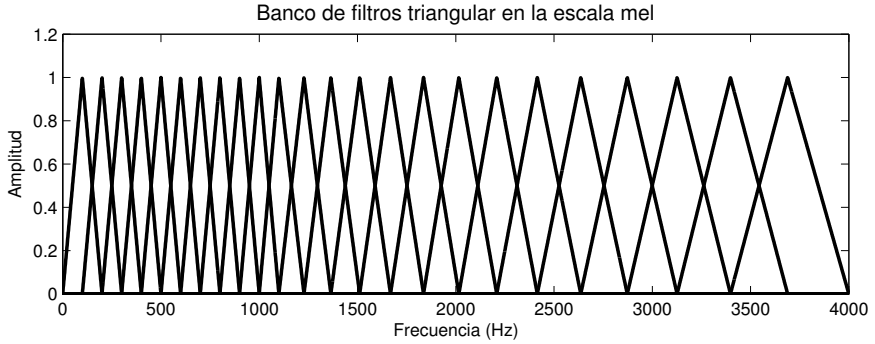


Figura 2: Banco de filtros en la escala MEL

Las características del baseline son calculadas en frames de 40 ms con ventana de Hamming solapadas un 50 %. Sea $x[n]$ la señal en un frame específico multiplicado por la ventana y $X[k]$ su DFT. Si además $H_l[k]$ es la transferencia del l-ésimo filtro MEL, la energía en esa banda está dada por:

$$E[l] = \sum_{k} |H_l[k]X[k]|^2,\ l = 0, \ldots, L-1,$$

donde L es el número de filtros MEL utilizado. Luego para encontrar los coeficientes ceptrales se calcula la DCT del logaritmo de la energía en las bandas:

$$MFCC[m] = \sum_{l=0}^{L-1} \log(E[l]) \cos\left(\frac{2\pi}{L} lm\right),\ m = 0, \ldots, M-1,$$

donde M es el número de coeficientes cepstrales [20]. Para poder describir la variaciones temporales de los coeficientes, también se calculan sus derivadas de primer y segundo orden ($\Delta MFCC, \Delta^2 MFCC$).

El coeficiente $MFCC[0] = \sum_{l=0}^{L-1} \log(E[l])$, resulta ser una medida de la potencia de la señal, por lo que en el baseline, se elimina para independizar la clasificación de esa medida. Por lo tanto en total se utilizan 19 $MFCC$, 20 $\Delta MFCC$ y 20 $\Delta^2 MFCC$, totalizando 59 características. Para calcular estas características se usa la librería *librosa* [21].

[2]Escala basada en la percepción auditiva. $MEL(f) = 1127.01048 \log\left(1 + \frac{f}{700}\right)$.

5. Modelos

5.1. Baseline - GMM

En la plataforma de trabajo se entrenan 7 modelos independientes para cada clase en una lógica de *uno contra el resto*. En el caso del baseline, para cada clase, se entrena dos modelos GMM, uno para las muestras pertenecientes a la clase y otro para el resto de las muestras. De esta forma, para cada clase ω_i se ajustan dos densidades de probabilidad con la siguiente forma:

$$p(\mathbf{x}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) = \sum_{k=1}^{K} \pi_k \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k),$$

donde K es el número de gaussianas, $\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k$ son la media y la matriz de covarianza de la componente k-ésima y π_k es el peso de cada componente. Se asume que las matrices de covarianza son diagonales. Entonces, sean $p_{\omega_i}(\mathbf{x}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$ y $p_{\bar{\omega}_i}(\mathbf{x}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$ las densidades modeladas para las muestras pertenecientes a la clase ω_i y las no pertenecientes respectivamente.

Para clasificar una muestra nueva, se calcula la log-probabilidad de que la muestra pertenezca a la clase ω_i (*positiva*) y se resta la log-probabilidad de la muestra perteneciente la clase $\bar{\omega}_i$ (*negativa*). Si el resultado de dicha resta supera cierto umbral γ se dice que la muestra pertenece a la clase ω_i.

Para implementar GMM, se utiliza la clase Gaussian Mixture[3] de *sklearn*[22] donde se pueden inicializar los parámetros utilizando k-means (por defecto) o al azar.

5.2. GMM con inicialización supervisada

Como primera aproximación al problema, se propone realizar una inicialización supervisada de los parámetros $\boldsymbol{\mu}_k$ y π_k. Se dividen las muestras pertenecientes cierta clase ω_i en siete conjuntos. El primer conjunto está formado por las muestras que sólo son de dicha clase, los seis restantes incluyen las muestras que son de esa clase y también de otra. Para fijar ideas, sean D_i los conjuntos de muestras de la clase ω_i. Debido al solapamiento temporal de eventos sonoros, una muestra $\mathbf{x}_k$[4] puede pertenecer a más de un conjunto D_i. De esta forma se dividen las muestras de entrenamiento de la clase ω_i en los siguientes conjuntos:

$$S_i^1 = \left\{ \mathbf{x} : \mathbf{x} \in D_i - \bigcup_{j \neq i} D_j \right\}$$

$$S_i^p = \left\{ \mathbf{x} : \mathbf{x} \in D_i \cap \left(D_p - \bigcup_{j \neq i,p} D_j \right) \right\} p = 2 \dots c,$$

donde c es el número de clases (7 en este caso). Se puede calcular los pesos de cada conjunto S_i^p como la cantidad de elementos sobre la totalidad de muestras:

$$\pi_p = \frac{\#S_i^p}{\sum_{j=1}^{c} \#S_i^j}.$$

Luego, para cada S_i^p se buscan $K\pi_p$ medias utilizando k-means. De esta forma las K componentes de la mezcla de gaussianas se buscan distribuir según la cantidad de muestras en cada conjunto.

[3] En el sistema original se utilizaba una clase denominada GMM que ha sido depreciada.

[4] En este caso, $\mathbf{x}_k$ es el vector de características correspondientes al *frame* k.

Básicamente, con este método se busca que la densidad de probabilidad de las muestras pertenecientes a la clase y la densidad de las no pertenecientes, modelen correctamente los solapamientos entre clases.

Para comparar el resultado de aplicar la inicialización supervisada y no aplicarla, se corren los sistemas para diferentes valores de γ. De esta forma se tienen diferentes puntos de trabajo $(ER, F1)$ que pueden graficarse en una curva de desempeño. Las curvas para ambos sistemas pueden verse en la Figura 3. Si se elige como punto de trabajo el punto que minimiza el error (y eventualmente maximiza F1) se tienen los valores de desempeño de la Tabla 2. Puede observarse que el resultado es ligeramente mejor, lo cual es razonable, ya que se está sumando información de la distribución de los datos en las clases.

Sistema	ER	$F1(\%)$
Baseline	0.85	39.0
GMM supervisado	0.82	40.7

Tabla 2: Resultados de GMM supervisado y no supervisado

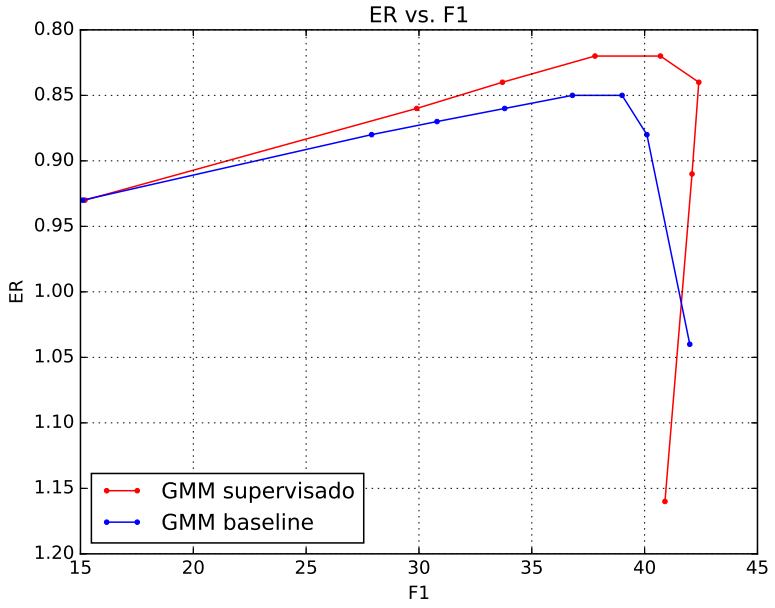


Figura 3: Curvas ER vs. $F1$.

Otra forma de comparar los resultados es utilizar las matrices de confusión. En este caso se miden las confusiones por segmentos de un segundo, de la misma forma que se presentó en la sección 3. Como es un problema de varias clases simultáneas, en un segmento, se cuentan las confusiones de cada clase contra el resto. De esta forma, la cantidad de aciertos o confusiones puede variar, por lo que se decide utilizar matrices de confusión normalizadas por filas. La Figura 4 muestras las matrices de confusión de ambos métodos. Se puede ver que al inicializar los parámetros se mejora en la detección de casi todas las clases y que las confusiones simplemente se redistribuyen.

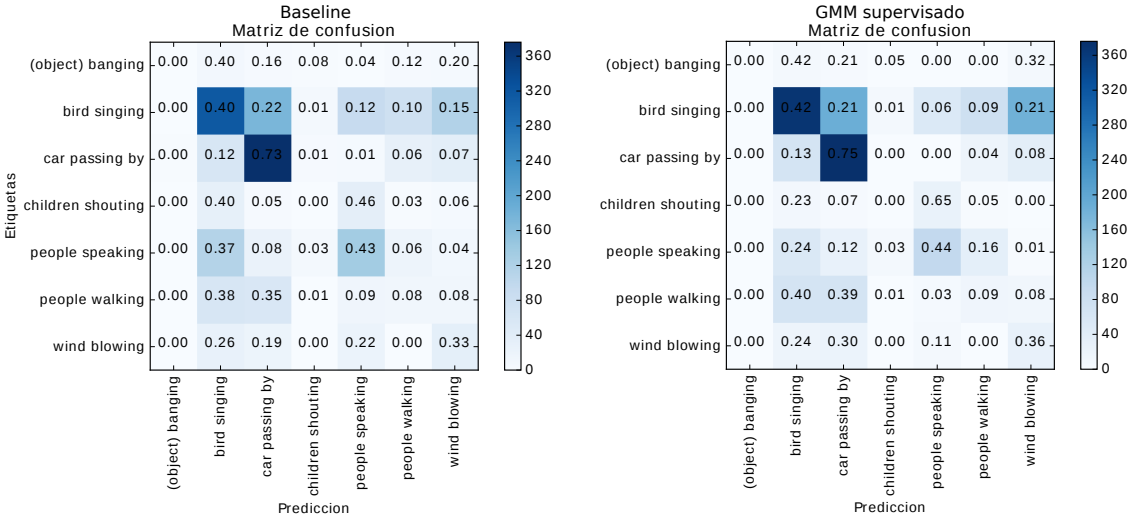


Figura 4: Matrices de confusión normalizadas.

5.3. Random Forest

Se prueba clasificar los datos utilizando Random Forest. En primer lugar se utilizan las mismas características que las secciones anteriores (MFCC y sus derivadas de primer y segundo orden). Se utiliza la implementación disponible en *sklearn* con 100 árboles. El resto de los parámetros se mantienen por defecto, por ejemplo, la cantidad máxima de características utilizadas para dividir los nodos es la raíz cuadrada del número de características ($\sqrt{N_{features}}$). En la Tabla 3 se pueden ver los resultados de aplicar este modelo que son muy similares a los del baseline aunque con un F1 mejor.

Sistema	ER	$F1(\%)$
Baseline	0.85	39.0
Random Forest	0.83	42.3

Tabla 3: Resultados de Random Forest con 100 árboles y parámetros por defecto

Como se entrenan modelos independientes para cada clase con la lógica de *uno contra el resto*, las muestras correspondientes a la clase *positiva* son muchas menos que las pertenecientes a la clase *negativa*. Por lo tanto, a la hora de dividir los nodos, el algoritmo le da más peso a la clase negativa. Para variar el punto de trabajo y encontrar el óptimo, se varía el peso que se asigna a la clase positiva en cada uno de los modelos en el parámetro *class_weight*, de forma de variar las probabilidades *a priori*. Con los parámetros por defecto, ambas clases tienen peso 1, lo que se debe hacer entonces es aumentar el peso de la clase minoritaria hasta el máximo dado por la relación de muestras entre las clases. De esta forma variamos el parámetro *class_weight* de la clase minoritaria con la siguiente función:

$$W = 1 + (M - 1)\alpha,$$

donde M es la relación entre la clase mayoritaria y la minoritaria y α varía entre 0 y 1. De esta forma, se busca que el peso que se quiere variar sea independiente de la cantidad de muestras.

Otra opción es recuperar las probabilidades *a posteriori* de que las muestras pertenezcan a la

clase positiva y negativa y restarlas para obtener una función de decisión:

$$g(\mathbf{x}) = P(\omega_p|\mathbf{x}) - P(\omega_n|\mathbf{x}),$$

donde la probabilidad $P(\omega_{p,n}|\mathbf{x})$ es la media de la probabilidad $P_k(\omega_{p,n}|\mathbf{x})$ de cada árbol k. La probabilidad $P_k(\omega_{p,n}|\mathbf{x})$ se calcula como la fracción de muestras que pertenecen a la clase $\omega_{p,n}$ en el último nodo luego de aplicar las decisiones. En el funcionamiento por defecto, se umbraliza $g(\mathbf{x})$, de forma tal que, si es mayor a cero, se decide que la muestra pertenece a la clase positiva. Para modificar el punto de trabajo, podemos variar dicho umbral tomando:

$$g(\mathbf{x}) \underset{\omega_p}{\overset{\omega_n}{\lessgtr}} \gamma.$$

Este método tiene como ventaja que no se debe entrenar un nuevo modelo para cada valor del umbral, ya que solo se utiliza en el momento de clasificar las muestras de *test*. El resultado de dicha prueba puede verse en la Figura 5 donde se compara con las curvas de los métodos presentados anteriormente. En el punto de trabajo óptimo, el resultado es igual al caso donde se utiliza GMM con inicialización supervisada ($ER = 0.82$, $F1 = 40.7\,\%$). En cuanto a la matriz de confusión (Figura 6) se puede ver que es diferente a las presentadas anteriormente. Principalmente en los puntos con más cantidad de muestras (BIRD SINGING y CAR PASSING BY) el resultado es peor. Por lo que, a pesar de que la curvas de desempeño parece mejor, no deja de ser un promedio de resultados. Viendo la matriz de confusión no se puede concluir que usar Random Forest sea mejor que GMM.

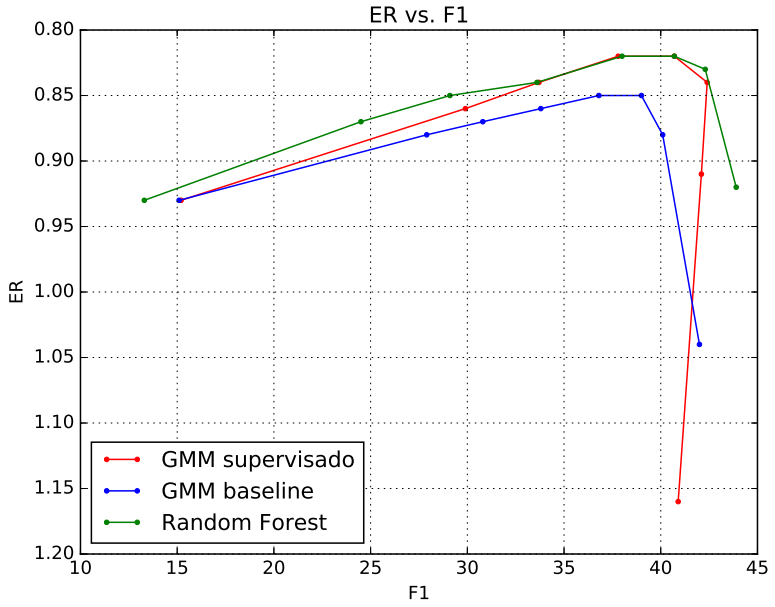


Figura 5: Curvas ER vs. $F1$.

Como en los casos anteriores, las mayores confusiones se dan entre CAR PASSING BY y BIRD SINGING que son las clases mayoritarias y en muchos casos se encuentran simultáneamente. Una forma de mejorar la separación entre esas clases es agregar características que las puedan diferenciar mejor. Como prueba simple se pueden agregar coeficientes MFCC, ya que en el entrenamiento de los árboles, Random Forest seleccionará las características que minimicen la impureza de los nodos.

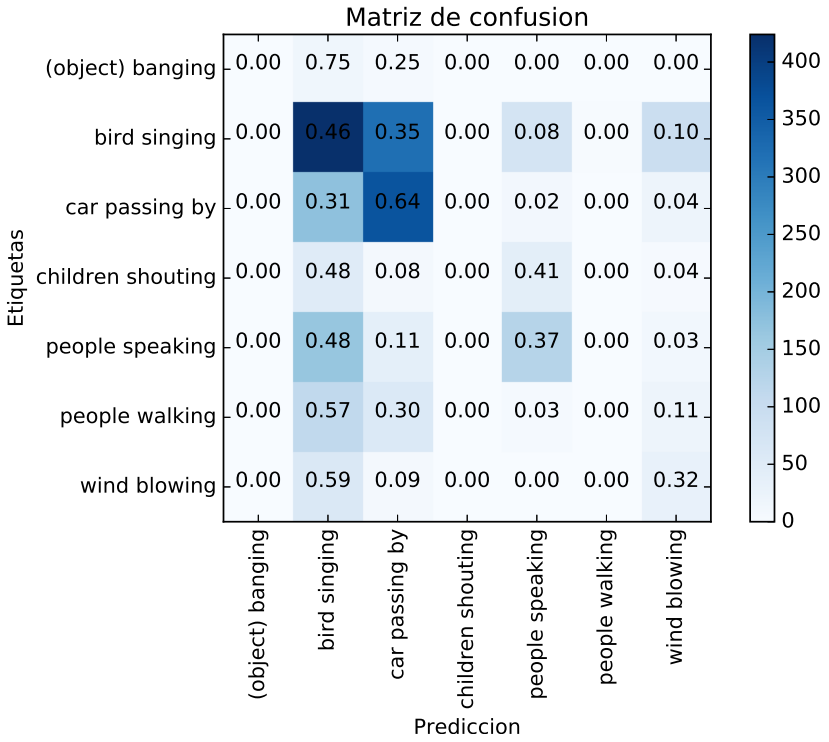


Figura 6: Matriz de confusión usando Random Forest.

Por lo tanto, se prueba mantener el coeficiente MFCC[0]. En la Figura 7 se compara la curva de desempeño contra las presentadas anteriormente mostrando una mejora en los resultados. También se puede notar, en la matriz de confusión de la Figura 8, que los errores mayoritarios disminuyen utilizando todos los coeficientes. El resultado mejora, ya que al agregar la información del MFCC[0], que describe la energía de la señal, los árboles logran discriminar los casos donde hay silencio (sólo ruido ambiente), no etiquetándolos. De esta forma se reduce considerablemente el número de falsos positivos.

Para confirmar la importancia de MFCC[0] para la clasificación, se obtiene de la clase Random Forest un ranking de las características más utilizadas en la construcción de los árboles. Si se grafican los resultados de las 60 características (Figura 9), se puede ver que el primer coeficiente es uno de los más relevantes para todos las clases. Otra conclusión interesante es, que en el caso de sonidos estacionarios (WIND BLOWING, CAR PASSING BY y BIRD SINGING), las derivadas de los MFCC no son muy relevantes para entrenar los árboles. En cambio, en los sonidos menos estacionarios (OBJECT BANGING, CHILDREN SHOUTING y PEOPLE WALKING), las derivadas pueden ser igual o más relevantes que los coeficientes estáticos.

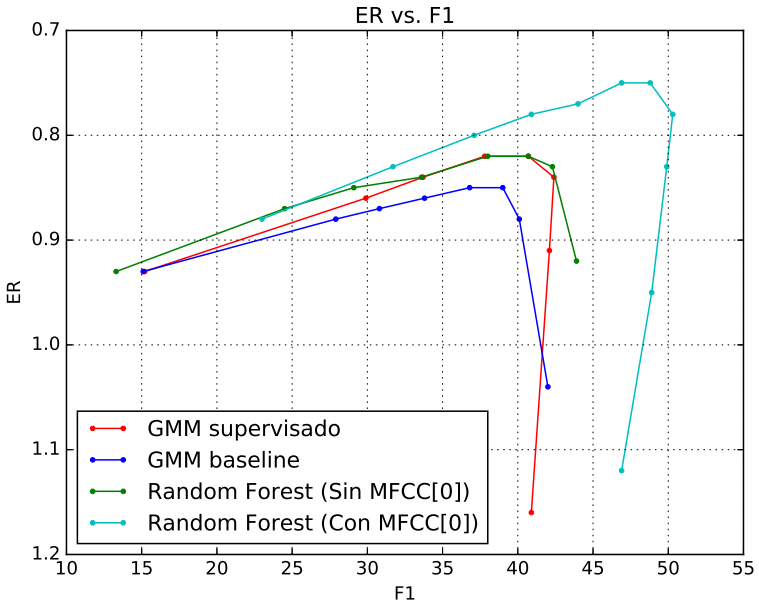


Figura 7: Curvas ER vs. $F1$.

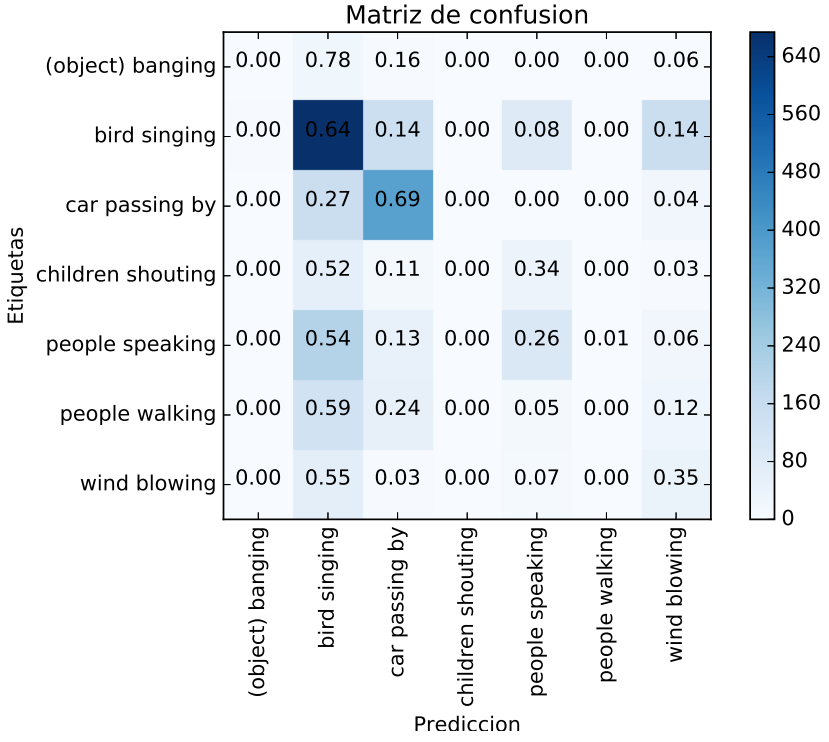


Figura 8: Matriz de confusión usando Random Forest los 20 MFCC y sus derivadas.

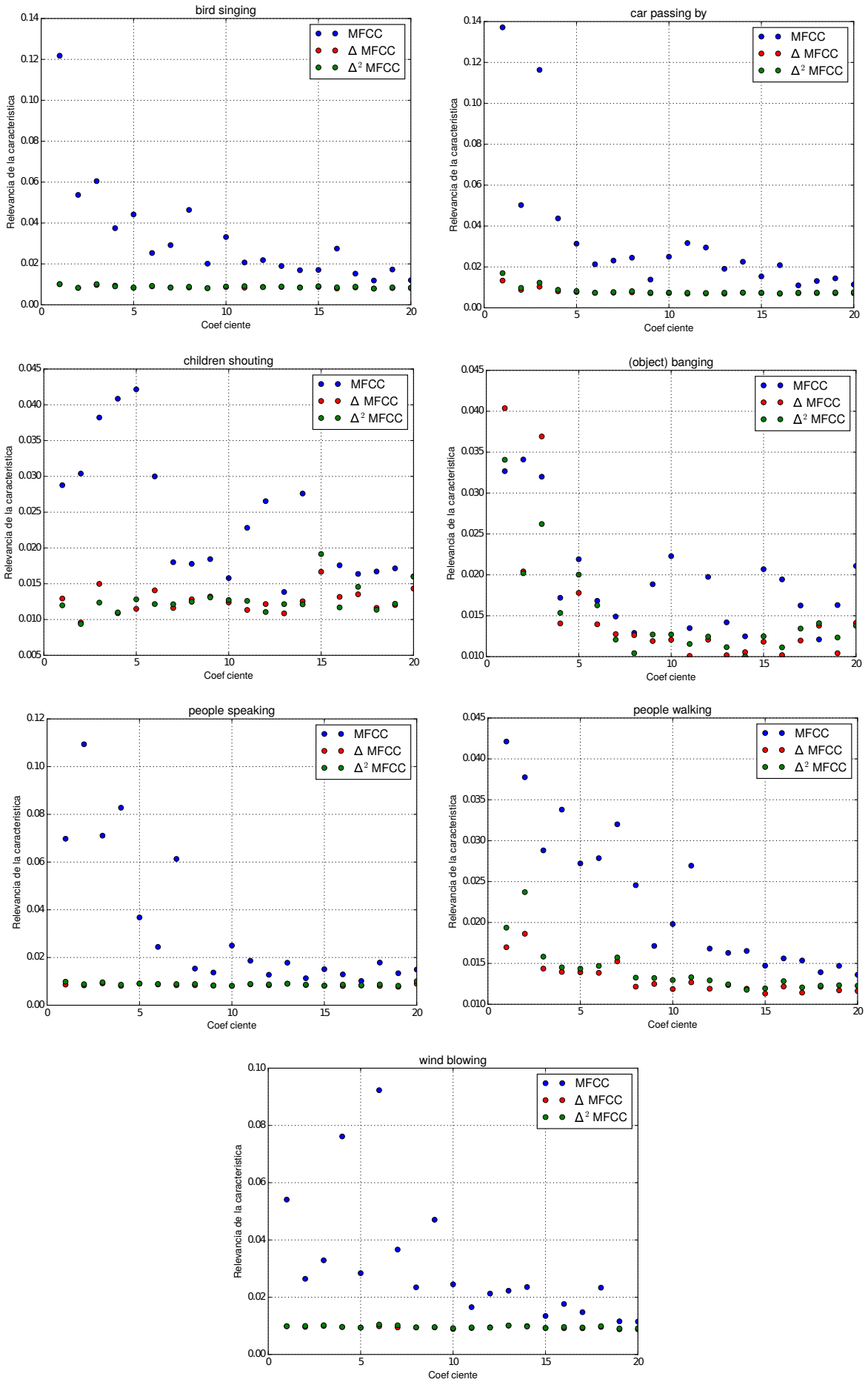
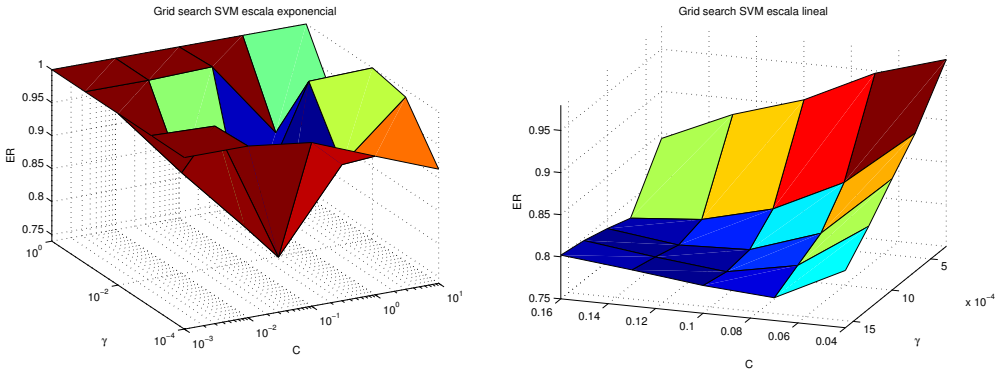


Figura 9: Relevancia de las características en la construcción de los árboles en Random Forest.

5.4. SVM

Se prueba el algoritmo de C-Support Vector Machines en su implementación de *sklearn* con un *kernel* RBF. Para ajustar los parámetros C y γ se hace una grilla de búsqueda y se busca minimizar ER. En primer lugar se busca en una grilla exponencial dada por $C = [10^{-3}, 10^{-2}, \ldots, 10^{1}]$ y $\gamma = [10^{-4}, 10^{-3}, \ldots, 10^{0}]$. El valor mínimo se da para valores de $C = 0.1$ y $\gamma = 0.001$ y corresponde al punto ($ER = 0.78, F1 = 42.67\,\%$). Luego, se busca en un entorno de esos valores en una escala lineal dada por $C = [0.03, 0.07, \ldots, 0.16]$ y $\gamma = [0.0003, 0.0007, \ldots, 0.0016]$ (ver Figura 10).



(a) Escala exponencial (b) Escala lineal

Figura 10: Grillas de búsqueda para SVM.

Una vez encontrados los parámetros ($C = 0.13$, $\gamma = 0.001$), se busca un punto de trabajo variando el umbral de la función de decisión dada por la distancia al hiperplano de separación. En la Figura 11 se puede ver que el resultado promedio es similar a Random Forest, pero viendo la matriz de confusión de la Figura 12 es claro que se obtiene un resultado peor. Por ejemplo, en las clases PEOPLE SPEAKING y WIND BLOWING se tienen muchos más errores. También se puede observar que las clases minoritarias se confunden mucho más con las clases mayoritarias. Esto puede deberse al hecho que SVM puede obtener modelos subóptimos cuando se tienen clases muy desbalanceadas como este caso [23].

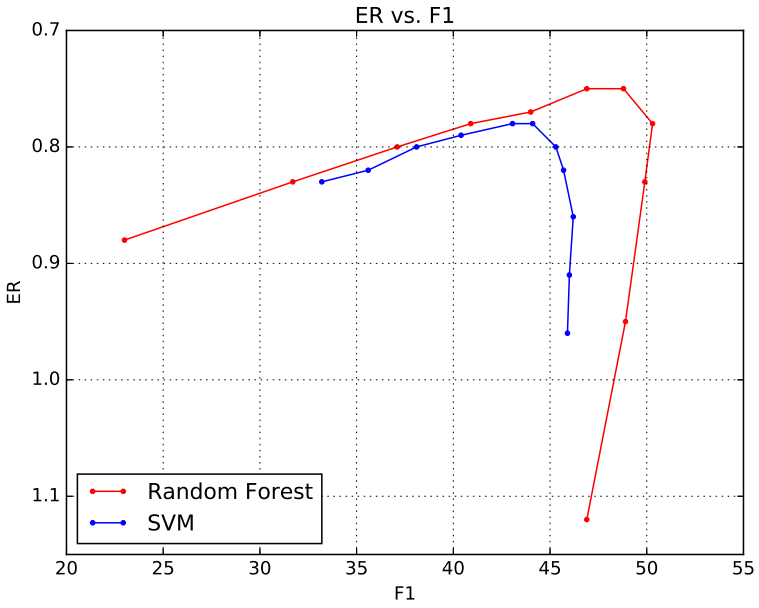


Figura 11: Curvas ER vs. $F1$.

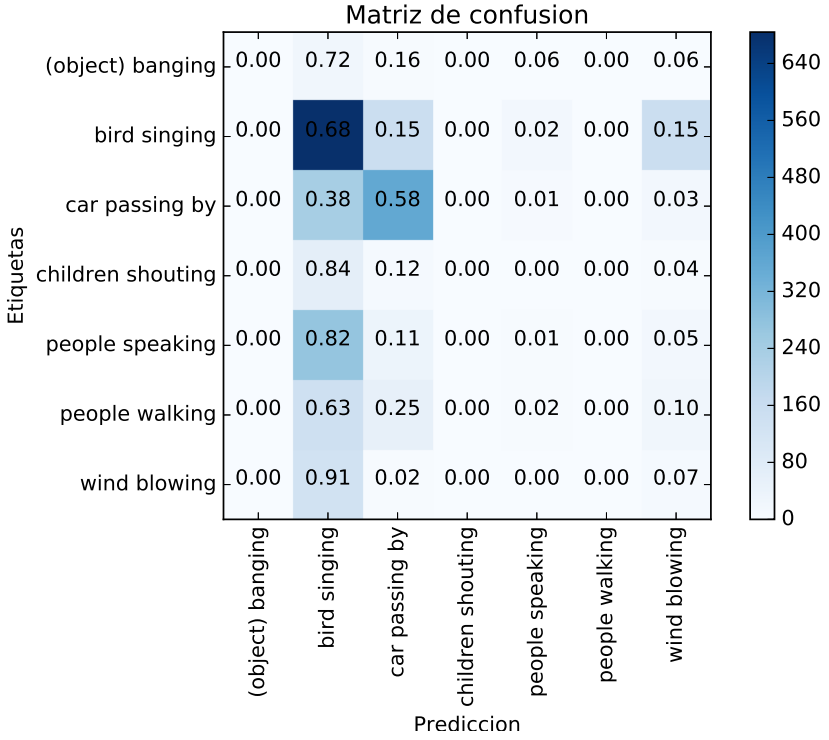


Figura 12: Matriz de confusión usando SVM

6. Edición de datos

6.1. Edición automática

Los resultados de las secciones anteriores muestran que el problema que se quiere atacar es difícil, principalmente debido a la superposición de eventos en el tiempo. Por otro lado, se puede ver que los resultados de todos los modelos presentados no difieren de forma contundente. Por lo tanto, se puede pensar que los errores cometidos son debidos a las propiedades de la base. Por ejemplo, el gran desbalance de las clases y algunos errores de etiquetado que se han encontrado. Para intentar solucionar el problema de los errores de etiquetado, se prueban métodos de edición automática de los datos. El primer método consiste en eliminar las muestras que cumplen que la mayoría de sus K vecinos más cercanos son de otra clase [24]. Para implementarlo se usa la librería *imbalanced-learn* [25]. Se prueba este método con Random Forest utilizando los mismos parámetros que anteriormente y la clase EditedNearestNeighbours (por defecto se trabaja con 3 vecinos). También se prueba el algoritmo presentado por Tomek [26], que es similar, pero con el cuidado de no eliminar muestras de la frontera. En la Figura 13 se compara la utilización de Random Forest con y sin edición de los datos. En el punto de trabajo óptimo se realizan la matrices de confusión de la Figura 14 donde se puede confirmar que el resultado es ligeramente peor utilizando k-NN y muy similar utilizando el algoritmo de Tomek.

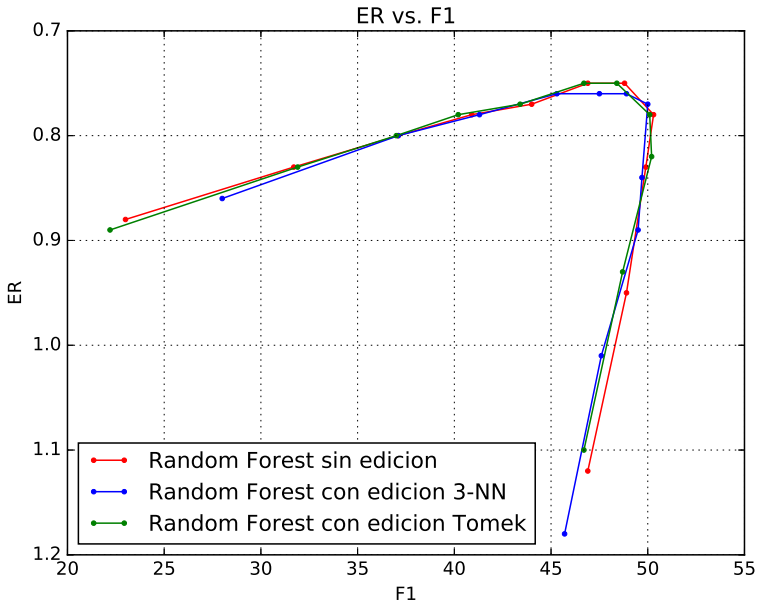
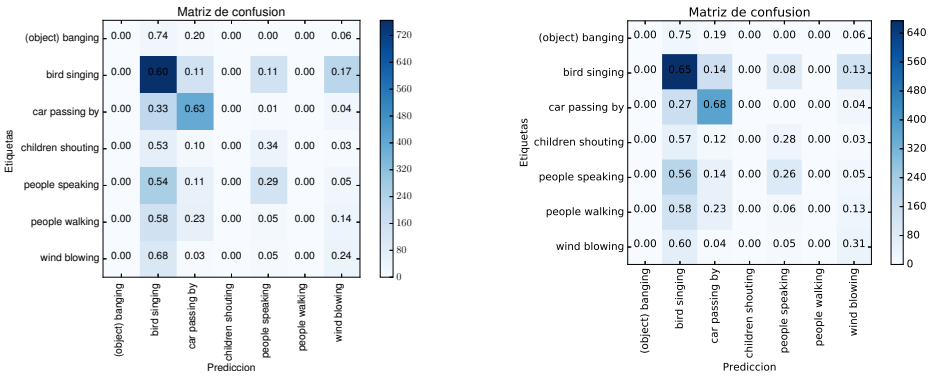


Figura 13: Curvas ER vs. $F1$.

6.2. Edición manual

Luego de examinar los datos manualmente, se confirma que el resultado anterior es coherente. Los errores de etiquetados son falsos negativos, o sea que hay zonas que deberían estar etiquetadas y no lo están. A la hora de entrenar, simplemente se tienen menos datos, aunque estos datos son correctos. El problema surge cuando se compara el resultado del sistema con el *ground-truth*, donde hay varios segmentos que el sistema detecta como falsos positivos y en verdad no lo son. Para



(a) 3-NN (b) Tomek

Figura 14: Matrices de confusión realizando edición automática de datos.

confirmar estos errores, se hace una revisión de la base de datos y se reentrena y testea el sistema. En las Figuras 15 y 16 se pueden ver las curvas de desempeño y la matriz de confusión luego de editar los datos manualmente. Se puede ver que la clase que mejora es BIRD SINGING ya que fue la clase que más se corrigió. Al mismo tiempo, las confusiones de otras clases con BIRD SINGING también aumentaron, pero el resultado global mejora. También se prueban los otros algoritmos utilizados (GMM, GMM supervisado y SVM) y en la Tabla 4 se presentan los resultados obtenidos. El único algoritmo que no mejora su resultado al editar los datos manualmente es GMM con inicialización supervisada.

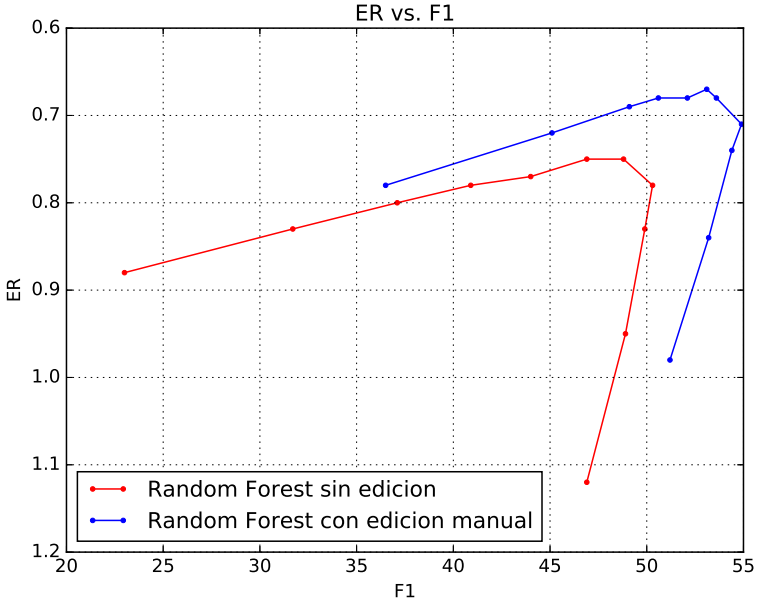


Figura 15: Curvas ER vs. $F1$.

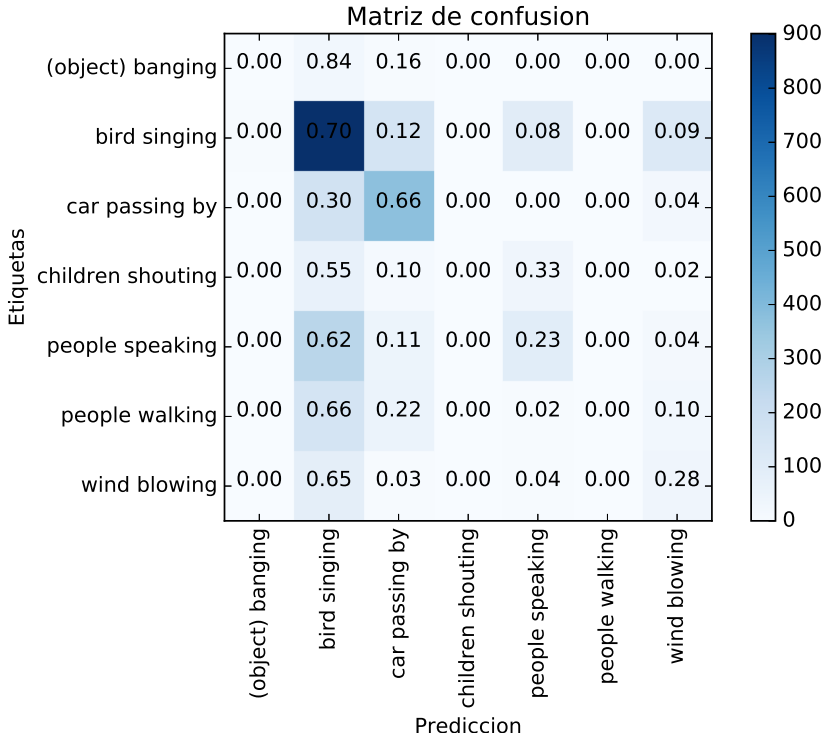


Figura 16: Matriz de confusión editando los datos manualmente.

	Sin edición de datos		Edición manual	
Sistema	ER	$F1(\%)$	ER	$F1(\%)$
Random Forest	0.75	48.8	0.67	53.1
SVM	0.78	44.1	0.70	51.5
GMM sup.	0.82	40.7	0.84	37.9
GMM baseline	0.85	39.0	0.79	42.5

Tabla 4: Resumen de resultados y comparación con y sin edición de datos

7. Tiempos de ejecución

Para comparar los tiempos de ejecución de todos los algoritmos, se corren en la infraestructura de cómputo de alto desempeño perteneciente a la Facultad de Ingeniería [27]. Para realizar una comparación fidedigna, se utilizan siempre 8 núcleos con 8 GB de RAM. En aquellas implementaciones que lo permiten (por ejemplo Random Forest) se especifica que paralelice el trabajo, utilizando todos los núcleos disponibles.

Sistema	Tiempo de train (s)	Tiempo de test (s)	Paralelización
Random Forest	311	78.8	Si
GMM baseline	387	76.9	No
GMM sup.	3390	80.5	No
SVM	15500	3000	No

Tabla 5: Tiempos de ejecución en el cluster de Fing

En la Tabla 5 se puede confirmar que Random Forest, además de arrojar buenos resultados de clasificación, es el sistema más rápido para entrenar y clasificar las muestras.

8. Conclusiones y trabajo futuro

Se probaron diferentes técnicas de reconocimiento de patrones en la base de datos TUT de eventos sonoros en áreas residenciales. Luego de ajustar los parámetros necesarios, se encontró que GMM con inicialización supervisada, Random Forest y SVM, dan mejores resultados que el baseline.

Se discutió el uso de $MFCC[0]$ como característica, ya que en el baseline no se utiliza con el objetivo de independizar la clasificación de la potencia de la señal y porque el resultado empeora al incluirlo (ER de 0.85 a 0.88). En cambio, cuando se utilizan otras técnicas como Random Forest o SVM, resulta ser de utilidad. Esto se debe a que los clasificadores podrían confundir los instantes de silencio con cualquier otra clase, generando falsos positivos.

Por último, se compararon los tiempos de ejecución corriendo los algoritmos en el cluster de Fing, encontrando que Random Forest es el más rápido. Entre otras cosas, esto se debe a que la implementación de *sklearn* permite paralelizar el entrenamiento de los árboles, aprovechando mejor los recursos.

Es interesante comparar los resultados con los trabajos que obtuvieron los mejores desempeños en DCASE 2016. La comparación se hace sobre los resultados de la base de desarrollo presentados en los artículos (Tabla 6). Aunque se tienen los resultados sobre la base de evaluación, no están disponibles las etiquetas, por lo que no se pueden obtener los resultados de los algoritmos aquí presentados.

Sistema	ER	$F1(\%)$
Random Forest	0.75	48.8
Gorin [15]	0.75	51.5
Adavanne [12]	0.79	42.1
Baseline [18]	0.85	39.0

Tabla 6: Comparación de resultados con los presentados en DCASE 2016

A lo largo del trabajo, se confirmó que el problema de detección y clasificación de eventos resulta difícil de resolver. En particular, se tiene una base de datos relativamente chica y muy desbalanceada, conllevando a que algunas clases nunca sean reconocidas. Al mismo tiempo, la base tiene etiquetas erradas, generando una evaluación incorrecta del sistema. En este sentido, se probaron técnicas automáticas de edición de datos que no mejoran el resultado, por lo que se decidió corregir las etiquetas manualmente.

Como trabajo futuro, se estudiará el uso de otras características. Aunque los MFCC han demostrado ser muy útiles, también se han utilizado la energía en las bandas MEL, medidas de frecuencia fundamental y características extraídas de la información estéreo [12]. En otro trabajo se utiliza el Banco de Filtros en 2D Gabor [9], recientemente diseñados para la clasificación automática del habla. Estas características han demostrado mejorar los resultados del estado del arte cuando se utilizan junto con MFCC, demostrando su complementariedad [28]. También se implementará una red neuronal recurrente (RNN-LSTM) que ha dado buenos resultados [12] y se probarán algoritmos de *subspace clustering* [29].

Referencias

[1] New York University. Página web del proyecto SONYC. `wp.nyu.edu/sonyc/`. [Internet; descargado 22-09-2016].

[2] Dirkjan Krijnders Daniel Steele y Catherine Guastavino. The sensor city initiative: cognitive sensors for soundscape transformations. En *Geoinformatics for City Transformations*, páginas 243–253. Technical University of Ostrava, January 2013.

[3] Audio research group of Tampere University of Technology. DCASE 2016 Challenge. `www.cs.tut.fi/sgn/arg/dcase2016/challenge`. [Internet; descargado 22-09-2016].

[4] Lawrence Rabiner y Ronald W. Schafer. *Digital Processing of Speech Signals*. Prentice Hall, New Jersey, 1978.

[5] Keith Dana Martin. *Sound-Source Recognition: A Theory and Computational Model*. PhD thesis, MIT. Cambridge, MA, 1999.

[6] Perfecto Herrera-Boyer, Anssi Klapuri, y Manuel Davy. chapter Automatic Classification of Pitched Musical Instrument Sounds, page 163–200. Springer, New York, 2006.

[7] K.S. Rao y S. Sarkar. *Robust Speaker Recognition in Noisy Environments*. Springer, 2014.

[8] Youngmoo E. Kim y Brian Whitman. Singer identification in popular music recordings using voice coding features. En *Interational Conference on Music Information Retrieval*, page 164–169, 2002.

[9] Benjamin Elizalde, Anurag Kumar, Ankit Shah, Rohan Badlani, Emmanuel Vincent, Bhiksha Raj, y Ian Lane. Experimentation on the dcase challenge 2016: Task 1 - acoustic scene classification and task 3 - sound event detection in real life audio. En *Detection and Classification of Acoustic Scenes and Events 2016*, 2016.

[10] Dmitrii Ubskii y Alexei Pugachev. Sound event detection in real-life audio. En *Detection and Classification of Acoustic Scenes and Events 2016*, 2016.

[11] Ying-Hui Lai, Chun-Hao Wang, Shi-Yan Hou, Bang-Yin Chen, Yu Tsao 1, y Yi-Wen Liu. Dcase report for task 3: Sound event detection in real life audio. En *Detection and Classification of Acoustic Scenes and Events 2016*, 2016.

[12] Sharath Adavanne, Giambattista Parascandolo, Pasi Pertila, Toni Heittola, y Tuomas Virtanen. Sound event detection in multichannel audio using spatial and harmonic features. En *Detection and Classification of Acoustic Scenes and Events 2016*, 2016.

[13] Matthias Zöhrer y Franz Pernkopf. Gated recurrent networks applied to acoustic scene classification and acoustic event detection. En *Detection and Classification of Acoustic Scenes and Events 2016*, 2016.

[14] Toan H. Vu y Jia-Ching Wang. Acoustic scene and event recognition using recurrent neural networks. En *Detection and Classification of Acoustic Scenes and Events 2016*, 2016.

[15] Arseniy Gorin, Nurtas Makhazhanov, y Nickolay Shmyrev. Dcase 2016 sound event detection system based on convolutional neural network. En *Detection and Classification of Acoustic Scenes and Events 2016*, 2016.

[16] Qiuqiang Kong, Iwnoa Sobieraj, Wenwu Wang, y Mark Plumbley. Deep neural network baseline for dcase challenge 2016. En *Detection and Classification of Acoustic Scenes and Events 2016*, 2016.

[17] Dai Wei, Juncheng Li, Phuong Pham, Samarjit Das, Shuhui Qu, y Florian Metze. Sound Event Detection for Real Life Audio DCASE Challenge. En *Detection and Classification of Acoustic Scenes and Events 2016*, 2016.

[18] Annamaria Mesaros, Toni Heittola, y Tuomas Virtanen. TUT database for acoustic scene classification and sound event detection. En *In 24rd European Signal Processing Conference 2016 (EUSIPCO 2016), Budapest, Hungary*, 2016.

[19] A. Mesaros, T. Heittola, y T. Virtanen. Metrics for polyphonic sound event detection. *Applied Sciences*, 6(6):162, 2016.

[20] S.B. Davis y P. Mermelstein. Comparison of Parametric Representations for Monosyllabic Word Recognition in Continuously Spoken Sentences. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 28(4):357–366, 1980.

[21] Brian McFee, Matt McVicar, Colin Raffel, Dawen Liang, Oriol Nieto, Eric Battenberg, Josh Moore, Dan Ellis, Ryuichi Yamamoto, Rachel Bittner, Douglas Repetto, Petr Viktorin, João Felipe Santos, y Adrian Holovaty. librosa: 0.4.1, October 2015.

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

[23] Rukshan Batuwita y Vasile Palade. *Imbalanced Learning: Foundations, Algorithms, and Applications*, chapter Class Imbalance Learning Methods for Support Vector Machines. John Wiley & Sons, 2012.

[24] D. Wilson. Asymptotic Properties of Nearest Neighbor Rules Using Edited Data. *IEEE Transactions on Systems, Man, and Cybernetrics*, 2:408–421, 1972.

[25] Guillaume Lemaître, Fernando Nogueira, y Christos K. Aridas. Imbalanced-learn: A Python Toolbox to Tackle the Curse of Imbalanced Datasets in Machine Learning. *CoRR*, abs/1609.06570, 2016.

[26] I. Tomek. Two modifications of CNN. *IEEE Transactions on Systems, Man, and Cybernetrics*, 6:769–772, 1976.

[27] S. Nesmachnow. Computación científica de alto desempeño en la Facultad de Ingeniería, Universidad de la República. *Revista de la Asociación de Ingenieros del Uruguay*, 61:12–15, 2010.

[28] M. Schädler, B. Meyer, y B. Kollmeier. Spectro-temporal modulation subspace-spanning filter bank features for robust automatic speech recognition. *The Journal of the Acoustical Society of America*, páginas 4134–4151, 2012.

[29] René Vidal, Yi Ma, y S.S. Sastry. *Generalized Principal Component Analysis*. Springer, 2016.