RECONOCIMIENTO DE PATRONES - 2016

# CLASIFICACIÓN DE SONIDOS EN ENTORNOS URBANOS

Daniel Aicardi - Gabriel De Cola

Proyecto Estándar

Universidad de la República, Uruguay

# Índice

# 1. INTRODUCCIÓN

El estudio de sonidos en entornos urbanos es de interés tanto para el registro como para la identificación de los mismos para, por ejemplo, caracterizar el paisaje sonoro de una ciudad. En particular, la detección de presencia de sonidos molestos o indeseables parece una buena aplicación.

Esta es un área de interés actual pero a la vez es un área en la que hay escasa investigación [1]. Uno de los mayores inconvenientes a la hora de estudiar los sonidos en entornos urbanos es la falta de datos etiquetados [1] debido al esfuerzo requerido para el etiquetado manual. Esto lleva a que hayan pocas bases y con pocos datos.

El otro problema que existe es la falta de un vocabulario común a la hora de clasificar los sonidos; es decir, la clasificación de sonidos en grupos semánticos puede variar de estudio a estudio haciendo difícil la comparación de resultados [1].

En [1] se propone una taxonomía (Figura (1) y la elaboración de una base de datos con el fin de promover un desarrollo común para investigación.

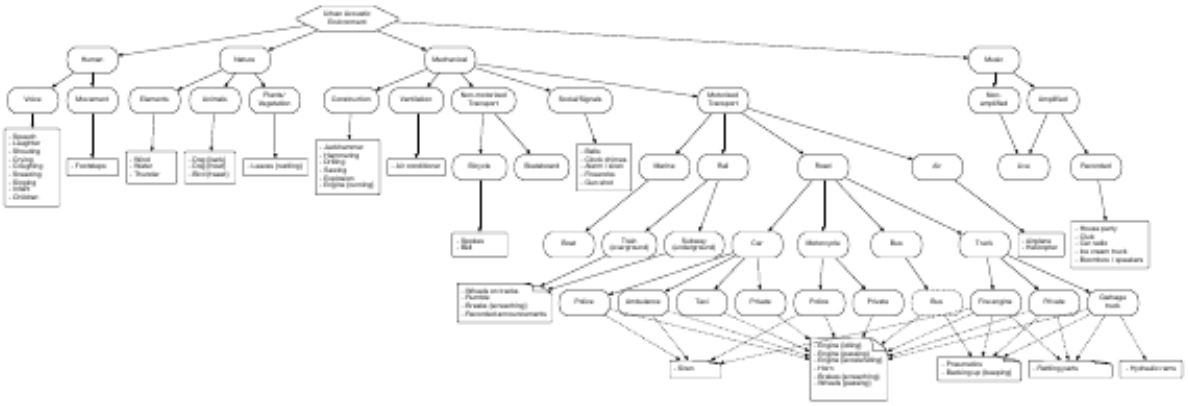


**Figura 1:** Taxonomía propuesta en [1]

## 2. BASE DE DATOS

Existen diferentes bases de datos creadas para la identificación de tipos de sonidos. Una de estas base es *UrbanSound8K*.
Esta base de datos, propuesta en [1], contiene trozos de grabaciones de sonidos en los que aparecen sonidos pertenecientes a un conjunto de 10 tipos de sonidos predefinidos:

0- aire acondicionado

1- bocina de auto

2- niños jugando

3- ladrido de perros

4- taladrado

5- motor moderando

6- disparo de arma de fuego

7- martillo neumático

8- sirena

9- música callejera

Esta base intenta asegurar, que los sonidos ocurran en entornos urbanos, todas las grabaciones sean tomadas en la zona donde ocurren y que la base de datos tenga suficiente cantidad y variedad de sonidos.
Las grabaciones son obtenidas desde un repositorio on-line llamado Freesound. Luego se chequean manualmente los audios comparando con los datos dados por el usuario que lo subió (metadatos). Se guardan los audios en los que se logra escuchar el sonido que se quiere encontrar.
Existen audios en los que aparecen varios sonidos a la vez y puede pasar que el sonido principal no sea el que buscamos pero en el fondo sí se escuche el que buscamos.
Los audios se dejan tal cual fueron subidos a Freesound.
Para investigación en identificación de fuentes sonoras, se toma un subconjunto de la base de datos anterior. A este subconjunto se lo llamó *UrbanSound8K* [1] y es el que se utiliza en

este trabajo.

Para crear *UrbanSound8K*, se asume que 4 segundos es suficiente para poder identificar un sonido particular por lo tanto se toman segmentos de audio de máximo 4 segundos. Así, si se tiene un audio de larga duración, se toman pedazos de 4 segundos obtenidos con ventana deslizante de 2 segundos.

Otro factor en consideración es que los datos ya están separados en 10 distintas carpetas ($folds$) que cumplen que dos segmentos de audio en distintas carpetas no provienen de una misma grabación. Ya que no se dispone de abundantes grabaciones, varios archivos de audio son segmentos de una misma grabación, pero para facilitar el posterior tratamiento se guardaron en una misma carpeta, por lo tanto entre carpetas no se comparten grabaciones.

En resumen, se cuenta con una base de datos de segmentos de audio identificados según una de las 10 clases predefinidas. Estos segmentos de audio tienen una duración máxima de 4 segundos.

## 3. DESCRIPCIÓN DEL PROYECTO

El lenguaje de programación elegido para este proyecto es *Python* ya que cuenta con bibliotecas específicas muy potentes para este tipo de problemas como lo son *numpy* y *scikit-learn* entre otros.
En principio se trabaja yendo por el camino seguido en [1] donde se obtienen las mismas características y se evalúan los mismos clasificadores aplicados. Luego se elige uno de estos clasificadores y se analiza la reducción de dimensión del problema mediante $PCA$. Se discute según distintas estrategias de clasificación.
También se hace un análisis profundo de la confiabilidad de la base de datos y se proponen distintas alternativas para filtrar los datos no confiables.

# 4. ELECCIÓN DE CARACTERÍSTICAS

Se eligió como conjunto de características, distintas propiedades estadísticas de los MFCC (Mel-Frequency Cepstral Coefficients) a lo largo del tiempo para los segmentos de audio. Estas características son las planteadas en [1].

En el procesamiento de audio, los MFCCs son coeficientes que representan las propiedades de la densidad espectral para tiempo corto, basados en una transformada lineal de coseno del logaritmo de la densidad espectral en una escala no lineal de frecuencia conocida como la *Escala Mel*, que se basa en la percepción subjetiva de la distancia entre tonos para distintos oyentes que interpretan una escala donde la distancia entre los tonos adyacentes es la misma.

Los MFCCs son bastante usados en el análisis de sonidos ambientales en el entorno urbano y han dado buenos resultados en términos de clasificación [1]. Para calcular los mismos se segmenta el dato de audio en varias ventanas de tamaño 23,2 ms con un 50 % de solapamiento entre ventanas conjuntas. Se trabaja con 40 bandas Mel distibuidas entre 0 y 22050 Hz y se conservan los primeros 25 coeficientes. Luego, el comportamiento para cada coeficiente a lo largo de todas las ventanas de audio se describió con las siguientes 11 propiedades aplicadas a cada MFCC obtenido: **mínimo**, **máximo**, **mediana**, **media**, **varianza**, **oblicuidad**, **kurtosis** y la **media y varianza de la primera y segunda derivadas de la señal**, resultando en un vector de características de dimensión 275. La obtención de estas características fue implementada en la plataforma *Matlab* tomando como base un script dado.

# 5. ELECCIÓN DE CLASIFICADOR

Para experimentar con distintos clasificadores se realizaron pruebas con validación cruzada de 10 carpetas, utilizando las mismas carpetas de la base de datos mencionadas previamente en el informe.
Se usaron los clasificadores: *SVM(rbf)*, *RandomForest500*, *KNeighborsClassifier* y *DecisionTreeClassifier*, y se midió el nivel de acierto de clasificación para cada carpeta y para cada clase, observando también su matriz de confusión correspondiente.

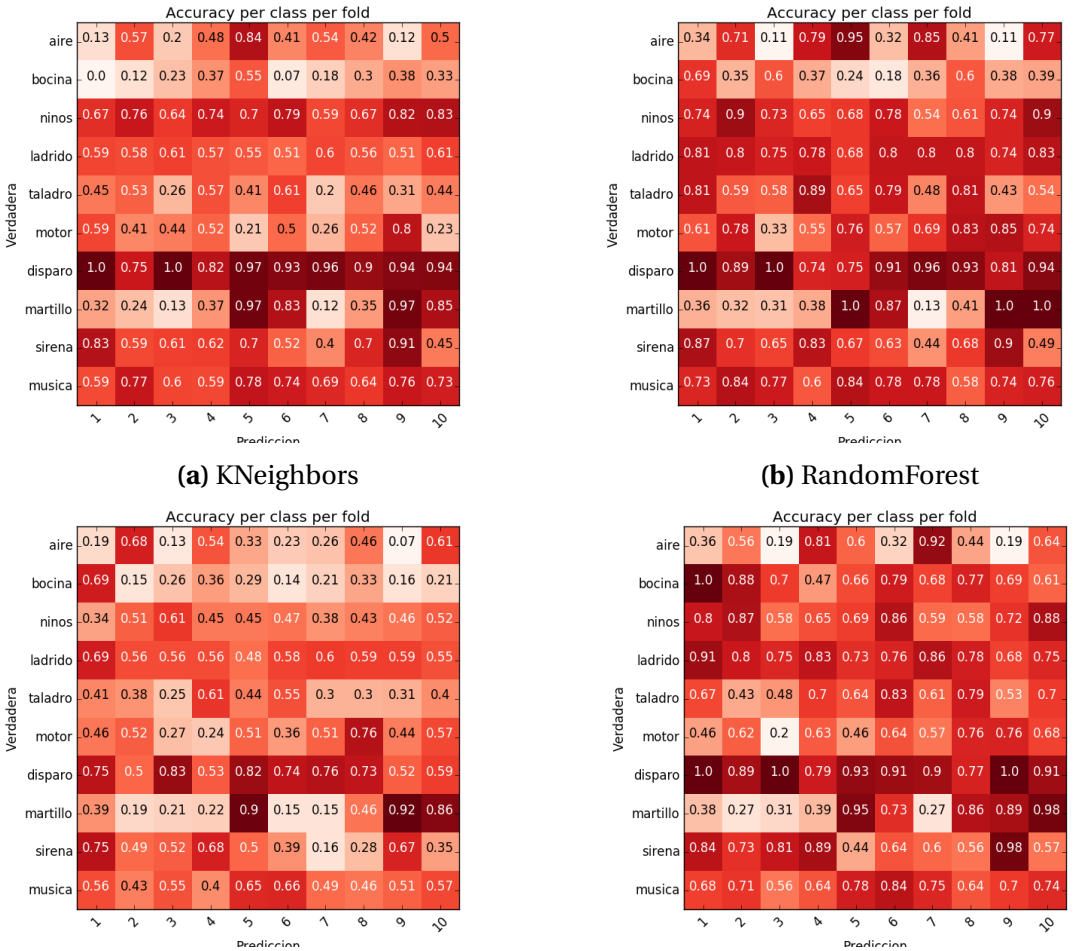
## 5.1. Clasificadores

Se corrió *C-SVM (rbf)* con $C = 10$ y $\gamma = 0{,}9x10^{-5}$ con un nivel de acierto de 66,3 %.
El método de *vecinos más cercano* se implementó para 4 vecinos ya que fue el que dio mejores resultados y se obtuvo un rendimiento de 55,1 %. Luego para *RandomForest*, en el cual se utilizaron 500 árboles entrenados con distintos subconjuntos de las muestras totales, se obtuvo un rendimiento de 66,2 %. El peor de todos resultó ser *DecisionTree*, que utiliza un único árbol para el clasificador, con un rendimiento de 46 %.
Los resultados para estos clasificadores obtenidos se pueden observar en la matriz de niveles de aciertos para cada $fold$ en cada carpeta de la Figura (2) y se ve claramente que en todos los clasificadores, se cumple que son siempre las mismas clases las que producen mayores errores y para las mismas carpetas. Por esto mismo, y observando el Cuadro 1, se decidió enfocar el resto del trabajo utilizando solo *SVM*.

**Cuadro 1:** Validación Cruzada

| Clasificador | Nivel de Acierto |
|---|---|
| SVM | 66,3 % |
| RandomForest | 66,2 % |
| KNeighbors | 55,1 % |
| DecisionTree | 46,0 % |

Cabe destacar que los niveles de acierto obtenidos son muy similares a los obtenidos en [1].




**(a)** KNeighbors **(b)** RandomForest

**(c)** DecisionTree **(d)** SVM

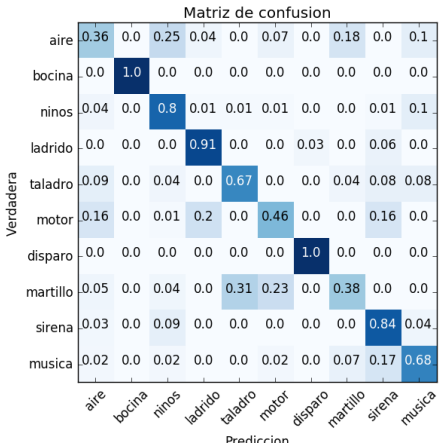
**Figura 2:** Rendimientos para cada fold en cada carpeta

## 5.2. SVM(rbf)

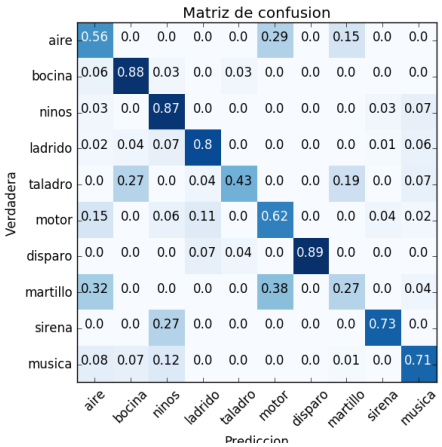
En nuestro caso se utilizó *C-SVM* con un *kernel rbf* ya que en general éste funciona para datos no linealmente separables. Para obtener el mejor rendimiento posible primero se optimizaron los parámetros de penalización $C$ y del coeficiente del *kernel* $\gamma$. Se implementó un algoritmo que realiza *GridSearch* de estos dos parámetros, utilizando una búsqueda exponencial para encontrar un entorno de los puntos ideales y luego una búsqueda lineal en ese entorno para seleccionar los mejores valores con la mayor exactitud posible. El algoritmo devolvió los siguientes valores: $C = 10$ y $\gamma = 0{,}9x10^{-5}$.

Luego se realizó la validación cruzada mencionada antes con el 66,3 % de nivel de acierto, y se obtuvieron las matrices de confusión para cada carpeta y total de las figuras (3) y (4).

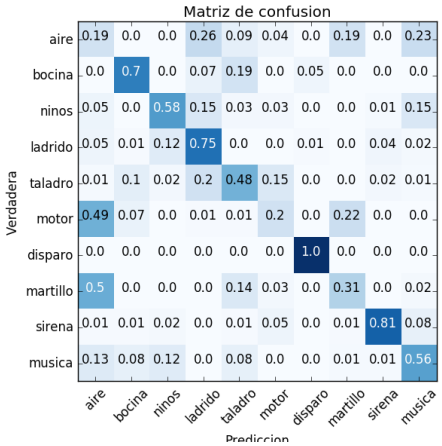
Otro detalle para SVM es que las estrategias para procesar datos de muchas clases, *one-*



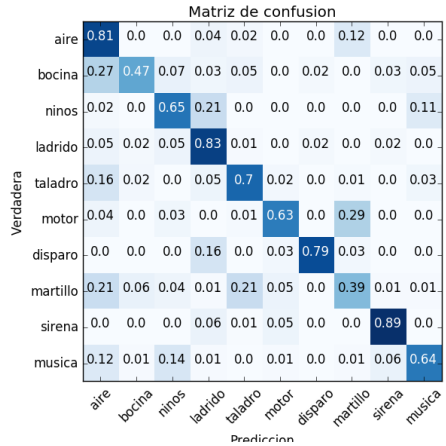

**(a)** Carpeta 1



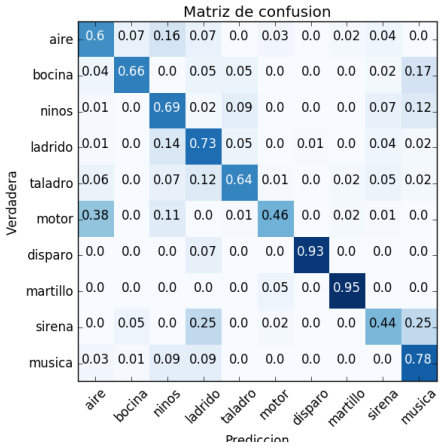

**(b)** Carpeta 2



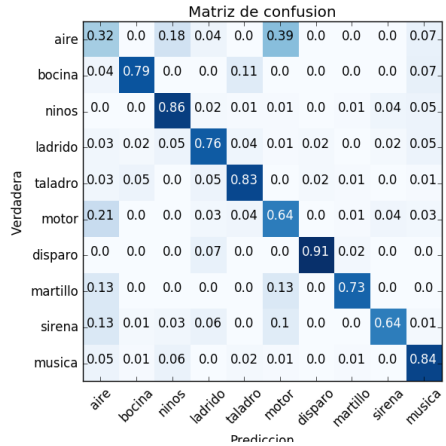

**(c)** Carpeta 3




**(d)** Carpeta 4

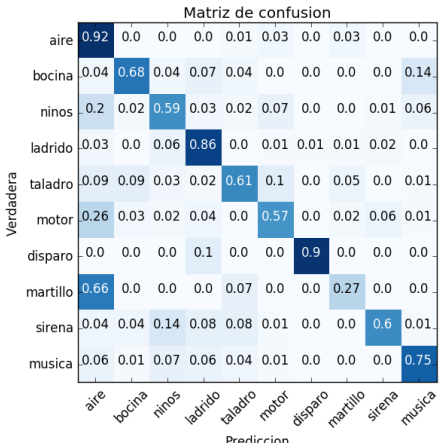



**(e)** Carpeta 5

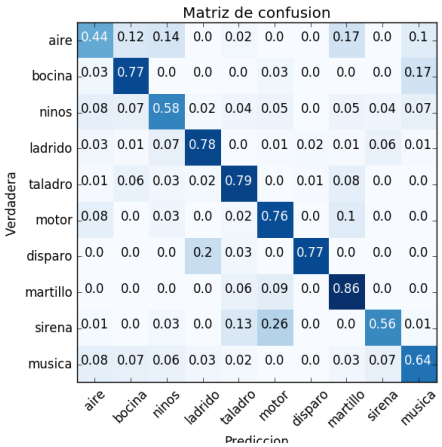



**(f)** Carpeta 6

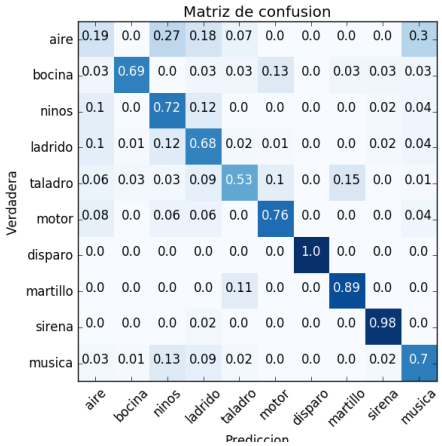
**Figura 3:** Matriz de Confusión para cada carpeta (1)



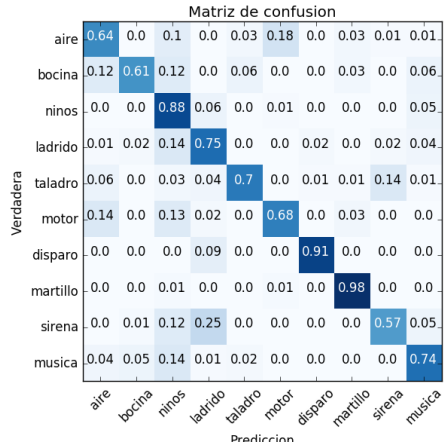

**(a)** Carpeta 7



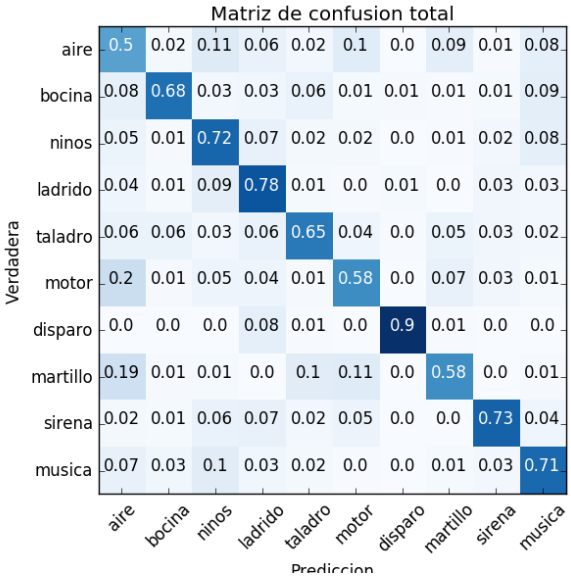

**(b)** Carpeta 8




**(c)** Carpeta 9




**(d)** Carpeta 10




**(e)** Total

**Figura 4:** Matriz de Confusión Total y para cada carpeta (2)

*vs-all* y *one-vs-one*, dieron los mismos resultados, por lo que no son un factor predominante en la construcción del clasificador y se adoptó la estrategia *one-vs-one* que es la opción por defecto.

Se puede ver en la Figura (4e) que las clases peor clasificadas son: *aire acondicionado*, *motor moderando* y *martillo neumático* con un nivel de acierto inferior al 60 %.
El *aire acondicionado* se confunde con casi todas las clases, *motor moderando* se confunde más que nada con *aire acondicionado*, y *martillo neumático* se confunde con *aire acondicionado* y *motor moderando*.
Por otro lado, *disparo con arma de fuego* alcanza un 90 % de nivel de acierto.

# 6. PCA

Debido a la cantidad de características obtenidas (275) y notando que la base de datos no tiene muchas datos, se puede presentar un problema de dimensionalidad. También se podría decir que 275 parecen demasiadas características para diferenciar entre 10 tipos de sonido. Se propone hacer un análisis por *componentes principales* ($PCA$) para analizar la dimensionalidad del problema.
Analizando la varianza acumulada en función de la cantidad de componentes principales de la Figura (5) se puede ver que para 40 componentes principales, la varianza acumulada es casi de 100 %.
En la Figura (6) se tiene un zoom de la figura anterior y se puede ver que con 5 componentes principales ya se está en el 95 % de la varianza acumulada total.
En la Figura (7) se puede ver que el nivel de acierto entra en una zona plana para alrededor de 23 componentes principales. Tomar 23 componentes principales parece razonable y se está bajando la dimensionalidad del problema diez veces, disminuyendo la complejidad del mismo.

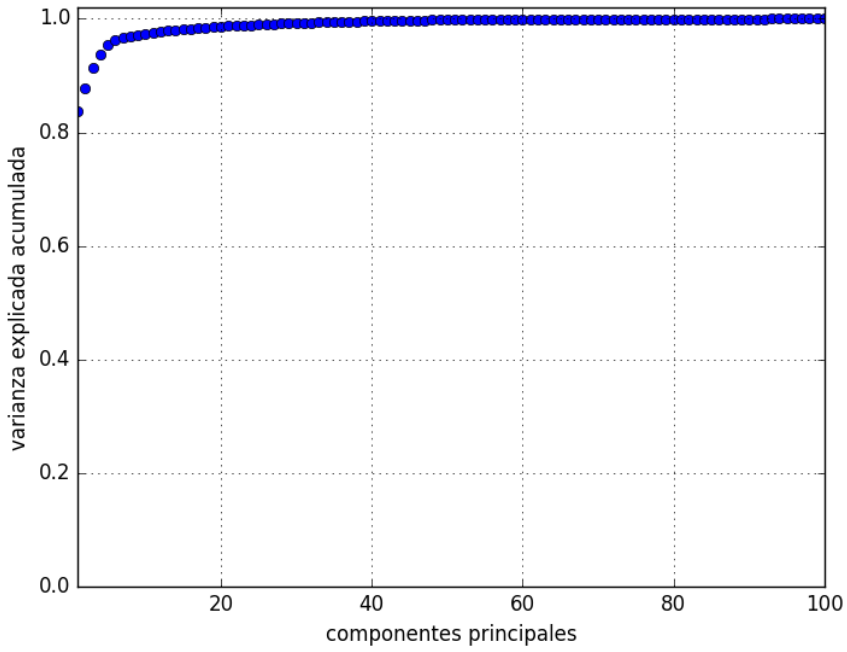



**Figura 5:** Varianza acumulada para los primeros 100 componentes principales

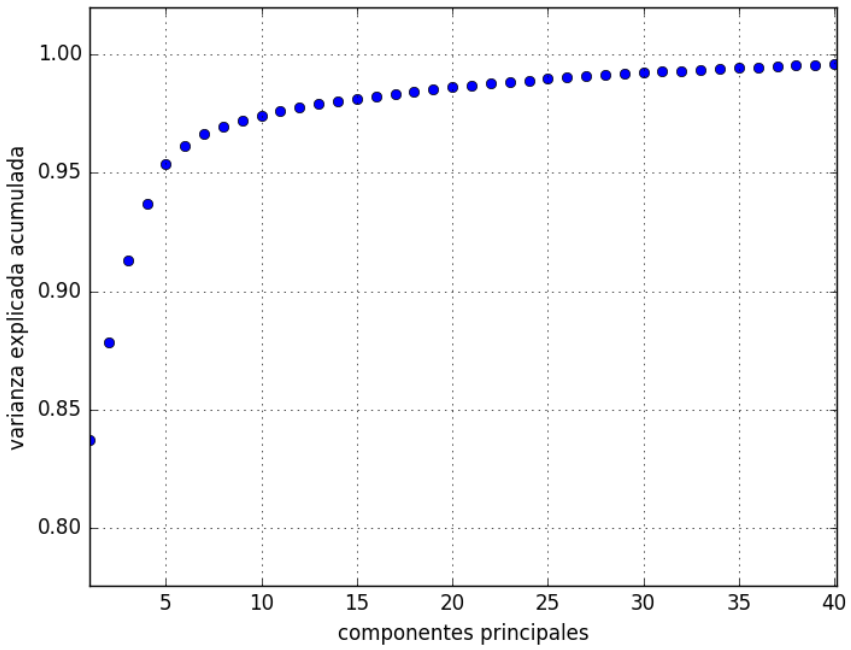



**Figura 6:** Varianza acumulada para los primeros 40 componentes principales

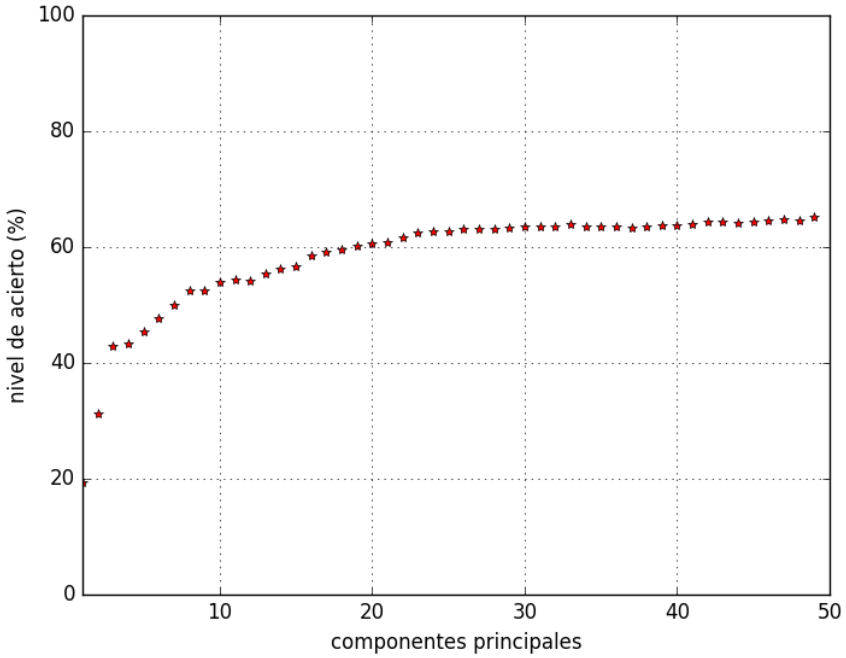



**Figura 7:** Nivel de acierto para los primeros 50 componentes principales

# 7. ANÁLISIS DE BASE DE DATOS

Para intentar mejorar los problemas de la clasificación recurrimos a analizar la base de datos a fondo ya que existen datos "malos"que están en alguno de los siguientes subgrupos:

1. Mal etiquetados
2. Clases mezcladas
3. Mala calidad de grabación
4. Audios muy cortos en duración
5. No se identifican con ninguna de las clases

Estos datos llevan a que el clasificador se confunda y para mitigar este efecto se adoptaron distintas estrategias para corregir la base de datos.

## 7.1. Eliminación de una carpeta

Primero se consideró un caso en el que una única carpeta pueda estar causando los mayores problemas para clasificar. Observando la matriz de rendimiento para cada carpeta y cada clase de la Figura (2), se puede concluir que la capeta con mayores errores en validación cruzada sería la tercera de las diez que hay. Por esto, se optó por directamente eliminar esta carpeta en la validación cruzada y trabajar con solo nueve carpetas. Al hacer esto se obtuvo un rendimiento del 66,5 %, o sea una mejora de dos décimas comparado al 66,3 % obtenido anteriormente. Este cambio es insignificante frente a lo que representa perder los datos de toda una carpeta para el clasificador final por lo que se recurrió a individualizar los datos clasificados incorrectamente.

## 7.2. Filtrado de los datos incorrectos

Para hacer esto simplemente se hizo una validación cruzada con las 10 carpetas y se observó qué datos eran clasificados incorrectamente. Ésta es una cantidad considerablemente grande (alrededor de 300 por carpeta) y ocurre que muchas de las instancias incorrectas, en realidad, por un oyente se consideran como una buen dato con etiqueta correcta.
Considerando esto, se prosiguió a ser más restrictivo con los datos eliminados.

## 7.3. Filtrado con restricción

En esta parte se consideraron solamente los datos que fueron clasificados con poca certeza o que haya habido una clasificación incorrecta por un gran margen.
Para poder hacer esto, se usó la opción *probability=True* para la función de clasificación. Con esta opción se puede ver el análisis de probabilidades de pertenencia a cada clase para cada muestra del conjunto de validación.
De esta forma, resolvimos que una muestra con alta probabilidad (mayor a cierto umbral) de pertenencia a una clase que no se corresponde con su etiqueta, es probable que esté mal etiquetada. Para esto se consideró un umbral de 0,6. Otra resolución fue que si una muestra tiene su máxima probabilidad por debajo de cierto umbral (0,4), esto significa que no es muy clara la clase de pertenencia y se puede suponer que hay clases mezcladas.
Estas muestras se eliminaron de la base original y se colocaron en dos carpetas diferenciadas para análisis posterior.

Ahora la cantidad de datos filtrados fue menor, por lo que el conjunto obtenido es más fácil de analizar escuchando. Se encontraron que la mayoría de los datos tienen los siguientes problemas enumerados al inicio.

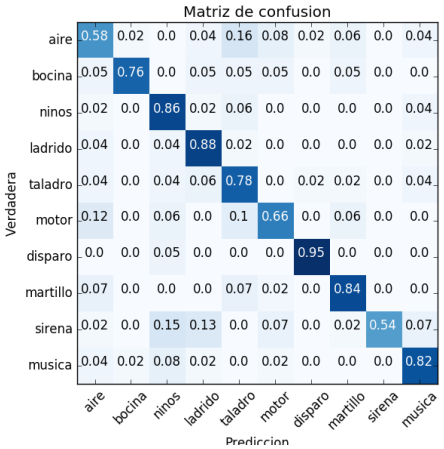
Considerando todo esto y aunque se tengan menos datos con los cuales entrenar se generó un clasificador con solamente el resto de las instancias de manera de probar el resultado en testing y discutirlo.

## 7.4. Clases problemáticas

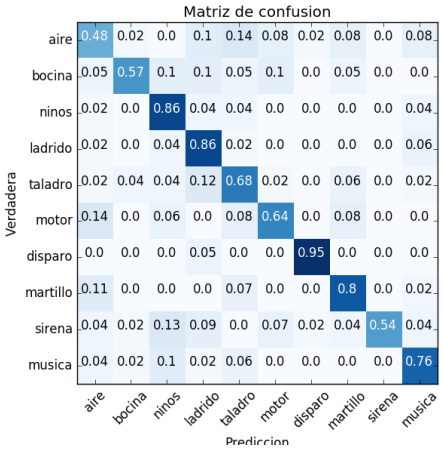
Volviendo a las matrices de confusión para *SVM*, se puede ver que las clases que brindan más problemas son el *aire acondicionado*, el *motor moderando* y el *martillo neumático*. Analizando los audios de estas clases se encuentran varios problemas de mal etiquetado o que no se identifica con certeza la clase a la que corresponde.
Además, la clase que genera mayoritariamente problemas es el *aire acondicionado* ya que todas las otras clases, y principalmente el *motor moderando* y el *martillo neumático*, se confunden con ella, debido a la gran variedad de tipos de grabaciones de la misma.

# 8. TESTING

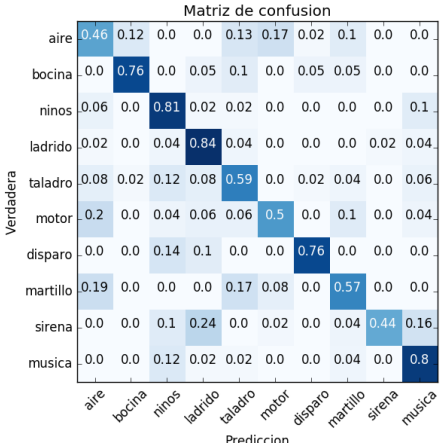
Para el test se tienen dos conjuntos *test1* y *test2* con características distintas pese a ser parte de la misma base de datos.
Los datos de *test1* son audios seleccionados sin tener en cuenta si esos datos son parte del conjunto de entrenamiento o no.
Los datos de *test2* son audios que no son parte del conjunto de entrenamiento.
El *clasificador1* es el clasificador entrenado con todos los datos de entrenamiento.
El *clasificador2* es el clasificador entrenado con los datos de entrenamiento que fueron filtrados con restricción.
Se testean los dos clasificadores previamente entrenados.



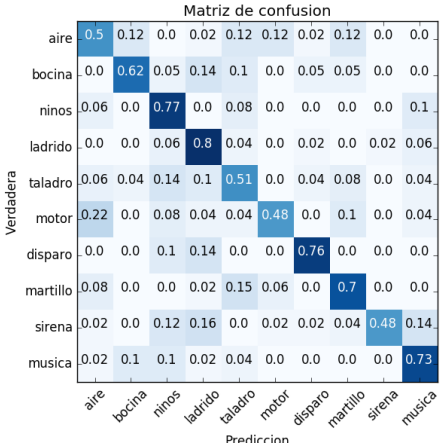

**(a)** Clasificador1 - datos de test1




**(b)** Clasificador2 - datos de test1




**(c)** Clasificador1 - datos de test2




**(d)** Clasificador2 - datos de test2

**Figura 8:** Matriz de confusión de los distintos clasificadores con los datos de test

En la Figura (8) se puede ver que para el conjunto *test1*, los tipos de datos mal clasificados coinciden para ambos clasificadores y son: *aire acondicionado* y *sirena*. Con la *sirena* confundiéndose principalmente con *niños jugando* y *ladrido de perro*.
Para el conjunto *test2* que es independiente del conjunto de entrenamiento, aparecen cuatro clases mal clasificadas, las peores son: *aire acondicionado, motor moderando* y *sirena*.
El *aire acondicionado* se confunde con *bocina, taladro, motor moderando* y *martillo neumático*.
El *motor moderando* se confunde mayormente con *aire acondicionado* y también con *martillo neumático*.
La *sirena* se confunde con *ladrido de perro, niños jugando* y *música callejera*.

Para ambos clasificadores los resultados son similares.

**Cuadro 2:** Comparación entre los distintos clasificadores para los conjuntos de test dados

| | *clas.1* | *clas.2* | *clas.1_PCA* | *clas.2_PCA* |
|---|---|---|---|---|
| *test1* | 75,6 % | 70,8 % | 74,5 % | 70,3 % |
| *test2* | 63,9 % | 62,8 % | 63,7 % | 62,8 % |

El resultado observado en el Cuadro 2 muestra que el nivel de acierto para ambos clasificadores, es más alto con los datos de *test1* respecto a los datos de *test2*. Esto se debe a que existen algunos datos de *test1* que fueron usados para entrenar los clasificadores.
La diferencia de nivel de acierto observada entre los clasificadores puede deberse a que el *clasificador2* fue entrenado quitando datos confusos y puede pasar que los datos de test sean de este estilo.
Para el conjunto *test2* se puede ver que la diferencia en el nivel de acierto entre los clasificadores es menor que para el conjunto *test1*.

Se puede decir que el *clasificador2* tiene un buen desempeño para datos de test independientes de los datos de entrenamiento.

En el Cuadro 2 también se puede comprobar que los resultados obtenidos para los clasificadores con 23 componentes principales es muy similar al desempeño de los clasificadores originales.

# 9. CONCLUSIONES

Se llegó a un resultado casi idéntico al propuesto en [1] con valores similares de nivel de aciertos para los distintos clasificadores.

También se encontró que el problema está sobredimensionado, ya que aplicando *PCA* a los datos y quedándose con menos del 10% de la dimensión de las características totales se llegó casi al mismo resultado. Esto da una idea de que muchas de las características son irrelevantes en el problema y que tal vez se puedan plantear algunas nuevas que describan el comportamiento evolutivo en el tiempo de los audios en vez de hacer utilizar propiedades que ignoran esto y se enfocan más en el comportamiento estático de la señal como las que utilizamos en este trabajo.

Analizando la base de datos se dedujo que existen varios errores presentes en la misma y que el problema de clasificación está limitado principalmente por este factor. Una solución para mejorar el problema posible sería eliminar la clase *aire acondicionado* que es donde existe más ambigüedad en los datos y que genera mayores conflictos de confusión entre clases. Además es una clase que difiere bastante de las otras en términos de intensidad y diferencia con el ruido general del ambiente urbano que la hace menos identificable en este entorno.
La *música callejera* si bien no dio grandes problemas, parece una clase demasiado general y podría ser dividida en subclases.

En etapa de testing, se dio lo que se esperaba con resultados similares para el conjunto de validación *test2* y mejores resultados para el conjunto de validación *test1* debido a las características del mismo. También se afirmó que el borrado de datos para generar un clasificador para el conjunto de *test2* no generó muchos cambios en el rendimiento del sistema ya que se eliminaron los datos "malos", los cuales no afectan la clasificación frente a datos totalmente nuevos, mientras que hubo una considerable baja de nivel de acierto para el conjunto de *test1* ya que con seguridad éste posee datos similares (de la misma grabación) a los datos borrados.

# REFERENCIAS

[1] SALOMON, J., JACOBY, C., AND BELLO, J.P. A Dataset and Taxonomy for Urban Sound Research. *Proceedings of the 22Nd ACM International Conference on Multimedia*