UNIVERSIDAD DE BUENOS AIRES
FACULTAD DE CIENCIAS EXACTAS Y NATURALES
Maestría en explotación de datos y descubrimiento del conocimiento

Año 2020 - 1er Cuatrimestre

APRENDIZAJE AUTOMÁTICO

TRABAJO PRÁCTICO Nº 2
TEMA: Algoritmos de ensambles aplicados en casos casi reales
FECHA: 17 de junio de 2020

INTEGRANTES:

Cisco, Nicolas
<ncis20@gmail.com>
Figueiro, Gabriel
<gabrielhfigueiro@hotmail.com>
Massaro Rocca, Patricio
<patomassaro@gmail.com>
Romero Victorica, Matías
<matiromerov@gmail.com>

***Resumen*—A pesar de los avances alcanzados en el descubrimiento del conocimiento, los métodos tradicionales de aprendizaje automático pueden no llegar a resultados satisfactorios a la hora de lidiar con conjuntos de datos complejos, los cuales están desbalanceados, poseen alta dimensionalidad o mucho ruido. Esto se debe a la incapacidad de los algoritmos para captar múltiples caracteristicas y la estructura subyacente de los datos. Los métodos de ensamble buscan fusionar los resultados obtenidos por un conjunto de algoritmos tradicionales, llamados aprendices débiles o *weak learners*, para lograr mejores métricas de predicción, utilizando esquemas de votación. El objetivo de este trabajo consiste en analizar los detalles de la implementación de estos algoritmos, aplicados a casos cercanos a la realidad. Más precisamente, se busca predecir el significado de un comando de voz que consiste en un número de 0 a 9.**

En primer lugar, se aplicaron transformaciones a los datos para reducir su dimensionalidad. Luego, previa separación de los conjuntos de entrenamiento, validación y prueba, se entrenaron distintos tipos de algoritmos, incluyendo naive Bayes, Random Forest, Gradient Boosting y Perceptrones multicapa. Los modelos obtenidos fueron probados en condiciones ideales y también reales, agregando ruido blanco y sonidos característicos de distintos ambientes, permitiendo el estudio del comportamiento de las métricas de performance en los distintos escenarios. Finalmente, se analizó la respuesta de los modelos utilizando audios grabados por los integrantes del grupo y se analizaron los resultados

I. Introducción

Los métodos de ensamble son procedimientos de aprendizaje estadístico que evocan el comportamiento social humano, basados en la búsqueda de conclusiones de distintos expertos antes de realizar una decisión. Esta idea de combinar opiniones ha sido utilizada incontables veces en la historia de la humanidad, probando que el criterio del grupo o comité es mejor que el de los individuos que lo componen, siempre bajo la premisa de que los individuos tienen una capacidad razonable [1]. De forma similar, los métodos de ensambles consisten en grupos de algoritmos de aprendizaje automático que fueron expuestos a diferentes porciones de los datos o poseen características distintas. Las decisiones de cada uno de los algoritmos se combinan entre sí bajo un criterio definido previamente. Un ejemplo muy simple consiste en una predicción binaria, en donde se entrenan varios algoritmos, y se decide la predicción con el método de mayoría simple.

Hoy en día estos métodos representan una buena parte de la investigación en aprendizaje automático, y son ampliamente utilizados en la industria tanto para clasificación como para regresión, en parte gracias al avance en la disponibilidad de dispositivos con gran poder de cómputo a costo razonable. Un ejemplo de uso de éstos métodos puede ser la interpretación de comandos realizados en forma vocal por un usuario, cuyo primer paso es clasificar las palabras escuchadas correctamente.

El objetivo del trabajo es la aplicación el estudio de métodos de ensamble y otros algoritmos para la clasificación de las palabras emitidas por el hablante. Los comandos emitidos consisten en números del 0 al 9, hablados en inglés. Por otro lado, los modelos obtenidos fueron puestos a prueba utilizando comandos con ruido agregado o grabados en entornos ruidosos para verificar su comportamiento.

En la sección II, se presenta una descripción del conjunto de datos, obtenidos del blog de Inteligencia Artificial de Google [2]. En la sección III se detallan las técnicas utilizadas a lo largo del trabajo para el procesamiento de los datos, el entrenamiento de los modelos y la generación de los distintos conjuntos de prueba. Los resultados y análisis se presentan en la sección IV. Finalmente, se muestran las conclusiones en la sección V.

II. Datos

Los audios para los conjuntos de entrenamiento, validación y testing fueron obtenidos del blog de Inteligencia Artificial de Google [2].

El set de datos consiste en 65.000 archivos de audio de un segundo de duración, grabados por cientos de personas diciendo 30 palabras cortas diferentes, de los cuales este trabajo solo utilizará los comandos que corresponden a dígitos.

III. Metodología

III-A. Conjuntos de entrenamiento, validación y prueba

En este problema en particular, los conjuntos de datos se encuentran previamente etiquetados para eliminar posibles sesgos en el algoritmo. Por ejemplo, los hablantes en el conjunto de entrenamiento son distintos que en el conjunto de validación o prueba. Realizar un *split* de forma aleatoria como se hizo en el trabajo anterior implica la posibilidad de que se haga validación utilizando los mismos hablantes que en el conjunto de entrenamiento, introduciendo una posible distorsión en las métricas de performance. Se generó un script que realiza la separación de los conjuntos utilizando las etiquetas presentes en los archivos.

III-B. Métrica de performance

Se decidió utilizar la exactitud o *Accuracy* como medida de performance debido a dos factores. En primer lugar, la naturaleza del problema implica buscar predecir correctamente la mayor cantidad de veces posible. Por otro lado, las clases en el conjunto de datos están balanceadas, escenario en donde la utilización de esta métrica es adecuada.

III-C. Extracción de atributos

El conjunto de datos consiste en varios archivos *.wav* con las órdenes grabadas. Dada la dimensionalidad de archivos de este tipo y para poder explotarlos con un algoritmo de aprendizaje automático, es necesario transformarlos.

Para lograr esto, se aplican varias técnicas con el fin de comprimir la información. En primer lugar se aplica la escala mel [3] a la transformada de Fourier de los datos, lo cual permite agrupar frecuencias. Luego, teniendo en cuenta la naturaleza de la voz humana y aplicando el *Cepstrum* [4], es posible obtener la estructura de los formantes (ver eq. 1).

$$Cepstrum(x) = \mathcal{F}(log(\mathcal{F}(x(t))) \tag{1}$$

Los coeficientes cepstrales en frecuencia mel (MFCCs) son calculados utilizando los espectros en escala mel, y usando transformada discreto coseno en lugar de DFT (transformada discreta de Fourier). Las características de interés, llamadas formantes, son las que definen la estructura de una palabra y permiten su clasificación. Estas residen en las primeras componentes del *Cepstrum*, por lo que se

utilizan los primeros 12 MFCCs y la energía, computando diferencias de primer y segundo orden, llegando a un total de 39 atributos. De estos atributos, se calculará su media y desvío estándar, reduciendo la dimensionalidad a 78 columnas.

Esta serie de transformaciones son realizadas utilizando el material suplementario provisto por la cátedra.

III-D. Entrenamiento y evaluación de los modelos

Se decidió entrenar los siguientes modelos:

- Naive Bayes (con distribución Gaussiana) [5]
- Random Forest [6]
- Gradient Boosting [7]
- Neural network [8] [9]

Para ello se utilizó la división del dataset provista en sets de entrenamiento, validación y prueba.

Primero se realizó la búsqueda de hiperparámetros entrenando con el conjunto de entrenamiento y validando la performance con el conjunto de validación. Esta búsqueda se hizo seleccionando combinaciones de hiperparámetros de manera aleatoria y para cada modelo se eligió la combinación que tenía mejor performance [10].

Posteriormente, se entrenó cada modelo utilizando el set de entrenamiento con los hiperparámetros seleccionados.

III-E. Pruebas en condiciones no ideales

Las audios grabados para el entrenamiento, validación y prueba del modelo fueron grabados en excelentes condiciones, éstas consisten principalmente en micrófonos de muy buena calidad y bajo ruido de fondo. Resulta interesante entonces evaluar los modelos entrenados en condiciones más cercanas a los casos reales, donde el micrófono utilizado es probablemente de un teléfono, en ambientes con ruido que pueden afectar fuertemente los componentes espectrales de la señales. Como se describe en la ec. 2, la señal obtenida es la suma de la señal original y los ruidos provenientes de condiciones no ideales.

$$y(t) = x(t) + \sum_{i=0}^{n} A_i \cdot \sigma_i(t) \tag{2}$$

En donde la señal obtenida, $y(t)$, es la suma de la señal de interés $x(t)$ y los ruidos σ_i, provenientes de distintas fuentes. A_i es un factor de amplitud utilizado en el experimento que permite aumentar o disminuir la relación señal-ruido (SNR).

En primer lugar, se agregó ruido blanco gaussiano utlizando la función *numpy.random.randn*. Previamente, fue necesario escalar el resultado entregado por *randn* teniendo en cuenta el máximo de la señal original.

Este tipo de ruido está presente en todo tipo de procesamiento de señales. Al agregarlo artificialmente, se recrean condiciones similares a cuando un usuario graba comandos desde un dispositivo. El ruido fue agregado en el campo temporal previamente al cálculo de los MFCCs.

El ruido blanco gaussiano aporta energía en todas las frecuencias, disminuyendo la SNR de la señal obtenida. Sin embargo, en el momento en que el usuario realiza un comando, los sonidos ambientales pueden agregar ruido en frecuencias clave para la clasificación del mismo. Para ilustrar el fenómeno, se eligieron sonidos ambientales de la calle, la cocina y un shopping [11] y se sumaron a las señales temporales de los audios para posteriormente procesarlas. Para que el problema sea lo más estocástico posible, se tomaron grabaciones de cinco minutos, y se seleccionó un segundo al azar para cada muestra del conjunto de entrenamiento.

Se realizó también un estudio de la sensibilidad de la métrica de performance a la constante A de la ec. 2. Esto permite ver su evolución a medida que la SNR disminuye.

Finalmente, se tomaron una serie de comandos en diversas condiciones de ambiente y de elementos de grabación. Esto permite observar el comportamiento de los algoritmos en condiciones reales, utilizando micrófonos de distintas tecnologías y calidades, agregando ruido blanco y de otro tipo tal como reverberación, sonidos de fondo, etc.

Algoritmo	**Performance**
Naive Bayes	0.52
Random Forest	0.54
Gradient Boosting	0.74
Neural network	0.74

Cuadro I: Performance de cada modelo para el conjunto de datos de pruebas

IV. Resultados

IV-A. Predicción del conjunto de pruebas

Como se puede observar en el cuadro I los modelos que lograron mejores resultados fueron las redes neuronales y gradient boosting. Aunque ambos tuvieron una exactitud global similar, si se comparan las matrices de confusión de cada uno (ver figura 1 y 2) se ve que cada modelo tiene sus particularidades a la hora de predecir cada número, por ejemplo, la red neuronal predijo considerablemente mejor el número tres que gradient boosting. En líneas generales ambos modelos tuvieron mejores predicciones para los números cinco y ocho, mientras fallaban más seguido para el uno y el dos.

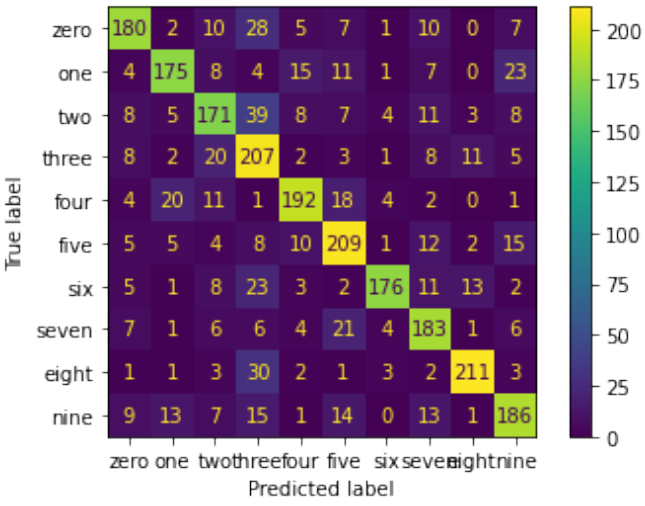

Figura 1: Matriz de confusión para el modelo de redes neuronales evaluado con el conjunto de testing

Naive bayes fue el modelo que tuvo las peores predicciones, seguido por Random Forest.

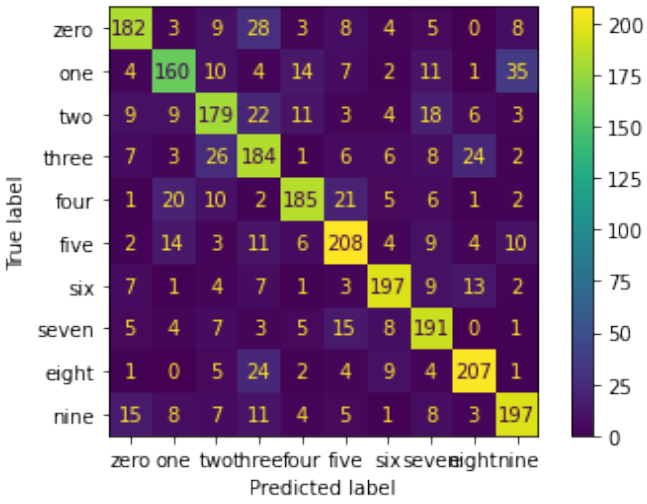

Figura 2: Matriz de confusión para el modelo de gradient boosting evaluado con el conjunto de testing

IV-B. Predicción en condiciones no ideales

En el caso del ruido blanco, se utilizo un valor de A de 0,01 hasta 0,1. Como se muestra en la figura 3, todos los algoritmos ven degradada su performance cuando A aumenta. Este resultado es intuitivo, considerando que la señal de interés se ve enmascarada. La reacción de los distintos algoritmos implementados es similar, aunque las pendientes de cambio y los puntos de partida son distintos. Tanto Naive Bayes como Random Forest llegan al límite del *random guess*, cuyo valor en este caso es de 0.1, dado que hay diez categorías distintas. El algoritmo de perceptrón multicapa logra la mejor respuesta, sin embargo, su sensibilidad al ruido no es despreciable.

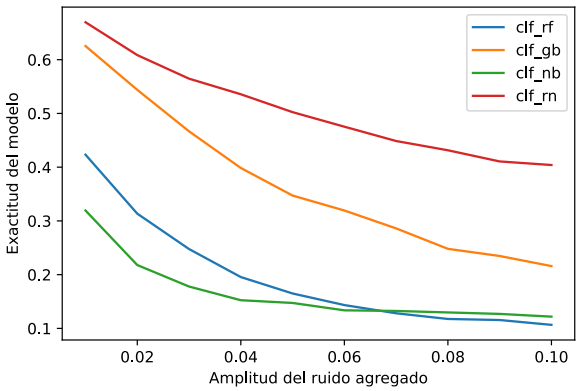

Figura 3: Exactitud de los modelos en función de la amplitud del ruido blanco agregado

En la figura 4 se muestra la reacción de los algoritmos frente a cambios en la amplitud del ruido ambiental A. Es importante notar que en este caso, A es mucho mas grande. Esto es debido a que los ruidos ambientales utilizados ya tienen incorporado el ruido blanco en su grabación, cuya potencia es baja debido a la calidad de grabación. Se pueden ver resultados similares al caso anterior, en donde todos los algoritmos ven degradada su respuesta aunque partiendo de valores iniciales y de pendiente distintos. Nuevamente, el algoritmo que mejor resultados ofrece es el del perceptron multicapa.

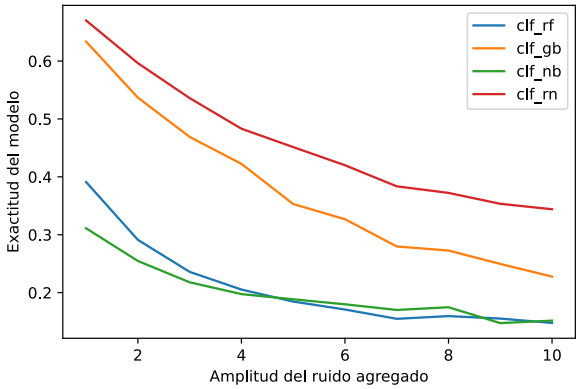

Figura 4: Exactitud de los modelos en función de la amplitud del ruido de calle agregado

Resulta interesante entonces ver si ruidos de distinta naturaleza pueden afectar en forma diferente a la performance de un algoritmo. Para la figura 5 se eligió el algoritmo del perceptrón multicapa, y se añadieron los distintos ruidos al conjunto de prueba. Se puede observar que en el entorno de un shopping el funcionamiento es mejor en todos los casos. Sin embargo, en el caso de la cocina y la calle se observa un cruce en las curvas. Teniendo en cuenta el método utilizado para agregar el ruido (ver III-E), es posible que en la calle aparezcan con mayor frecuencia ruidos fuertes que afectan los componentes espectrales de la señal, mientras que en la cocina los ruidos suelen ser constantes y de menor volumen.

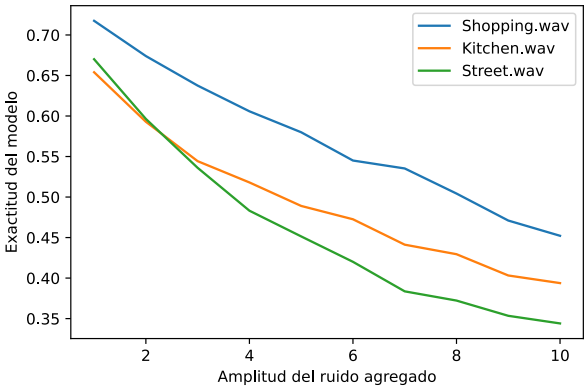

Figura 5: exactitud del modelo perceptron multicapa en función de la amplitud para ruidos de distintos entornos

Además, se evaluaron los modelos con señales de audio grabados en forma casera, es decir, con dispositivos hogareños y en ambientes ruidosos. Los resultados de estas evaluaciones se pueden ver en la figura 6.

Los conjuntos de datos utilizados para dicha evaluación presentan las siguientes características:

- **calle**: grabado con celular en calle poco transitada.
- **hombre**: grabado con un celular en ambiente cerrado con ruido muy leve de fondo.

- **hombre-ruido**: grabado con un celular con ruido de canilla abierta
- **mujer**: grabado en una computadora personal.
- **mujer-ruido**: grabado con un celular al lado de una cocción de carne a las brasas.
- **música**: grabado con celular al lado de un parlante reproduciendo música a volumen moderado.
- **nene**: grabado con un celular en el que el hablante es un niño de seis años.
- **nene-ruido**: grabado con un celular al lado de una cocción de carne a las brasas por un niño de seis años.
- **normal**: grabado con celular en ambiente cerrado sin ruido de fondo.
- **ventilador**: grabado con celular al lado de un ventilador.
- **test**: conjunto de testing del dataset
- **test-shop**: conjunto de testing del dataset con ruido de shopping agregado
- **test-cocina**: conjunto de testing del dataset con ruido de cocina agregado
- **test-camion**: conjunto de testing del dataset con ruido de trafico agregado
- **test-gaussian**: conjunto de testing del dataset con ruido gaussiano (0.05) agregado

Se puede observar que la mayoría de los modelos cuando se corren de las condiciones de entrenamiento tienden a tener una caída significativa en su performance. Un ejemplo de ellos es la evaluación con voces de nenes o cerca de un ventilador.

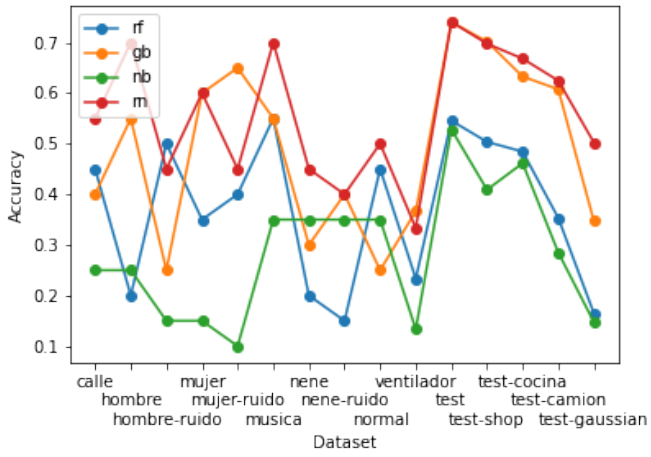

Figura 6: Performance de los modelos en función de los datasets poco ideales

V. Conclusiones

Durante el transcurso del trabajo, se aplicaron conceptos teóricos relacionados a métodos de aprendizaje automático como ensambles y redes neuronales a un caso práctico, utilizando herramientas y librerías disponibles en *Python*. Se logró entrenar distintos modelos que clasifican comandos de voz maximizando una métrica de performance elegida en base a la problemática planteada. Estos modelos fueron puestos a prueba utilizando muestras expuestas a distintos escenarios, en condiciones apartados del caso ideal.

Aunque el algoritmo de redes neuronales podría ser considerado el mejor y más robusto en términos de performance, cada modelo podría tener su uso dependiendo de las necesidades del problema que se quiera resolver.

En principio, Naive bayes podría ser utilizado como una primera aproximación debido a que no necesita una selección de hiperparámetros, lo que hace muy fácil su aplicación. Por otro lado, no es demandante computacionalmente. No hay que dejar de lado que estas ventajas se ven opacadas a que resultó ser el algoritmo que peor predijo de entre los analizados.

Random Forest y Gradient boosting producen mejores resultados, el segundo significativamente mejores, con el costo de tener que realizar una búsqueda de hiperparámetros y un mayor requerimiento computacional.

El agregado de ruido en forma artificial degradó la respuesta de los algoritmos. Esto resulta intuitivo, ya que al utilizar señales más ruidosas que las utilizadas en el entrenamiento, las métricas de performance empeorarán. De todas maneras, es importante notar que las respuestas de los modelos tuvieron una forma similar cuando se varió la amplitud del ruido A. Algunos de ellos, partiendo de exactitudes mas acotadas, llegaron al límite del *random guess*, con lo cual su poder predictivo es nulo. Todos los algoritmos tuvieron una respuesta pareja al introducir ruidos provenientes de distintos ambientes.

Finalmente, se debe resaltar que en líneas generales, los algoritmos tuvieron una performance pobre para los audios grabados por los integrantes del grupo. Este suceso puede ser atribuido a que los mismos difieren significativamente del conjunto de datos utilizado para entrenar a los modelos. Este tipo de grabaciones caseras posee varios factores externos muy difíciles de controlar que introducen distorsiones mayores que en los casos con ruido agregado artificialmente. Esto resulta en una respuesta mucho más errática en los algoritmos, incluso frente a condiciones de grabación similares.

Referencias

[1] Matteo Re and Giorgio Valentini. *Ensemble methods: A review*, pages 563–594. 01 2012.

[2] Kaggle. Speech commands dataset. https://ai.googleblog.com/2017/08/launching-speech-commands-dataset.html, 2017.

[3] S. S. Stevens, J. Volkmann, and E. B. Newman. A scale for the measurement of the psychological magnitude pitch. *The Journal of the Acoustical Society of America*, 8(3):185–190, 1937.

[4] Stan Z. Li and Anil Jain, editors. *Cepstrum Transform*, pages 180–180. Springer US, Boston, MA, 2009.

[5] scikit-learn developers. Gaussian naive bayes (gaussiannb). https://scikit-learn.org/stable/modules/generated/sklearn.naive_bayes.GaussianNB.html, 2019.

[6] scikit-learn developers. Random forest classifier. https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html, 2019.

[7] scikit-learn developers. Gradient boosting classifier. https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.GradientBoostingClassifier.html, 2019.

[8] Google. Module: tf. https://www.tensorflow.org/api_docs/python/tf.

[9] Google. Module: tf.keras.layers. https://www.tensorflow.org/api_docs/python/tf/keras/layers.

[10] scikit-learn developers. Parameter sampler. https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.ParameterSampler.html, 2019.

[11] Amir Anhari. A database of multichannel environmental noise recordings. https://www.kaggle.com/aanhari/demand-dataset?, 2018.