Proyecto Final de Aprendizaje de Máquina

Music Genre Classifier

Integrantes:
Leandro Rodríquez Llosa C-41X

Niley González Ferrales. C-411

Arian Pazo Valido. C-311

Cuarto año. Ciencias de la Computación.
Facultad de Matemática y Computación, Universidad de La Habana, Cuba

Junio 2023

Resumen

TODO

Palabras clave — aprendizaje automático · clasificación · géneros musicales · transformada wavelet discreta · transformada wavelet compleja de doble árbol

I. Introducción

TODO

La clasificación de géneros basada en la Transformada de Fourier, utilizando MFCC y espectrogramas, se ha explorado con éxito en los últimos años. Aunque la Transformada de Fourier tiene una alta resolución en el dominio de la frecuencia, tiene una resolución cero en el dominio del tiempo. Esto significa que puede decirnos exactamente qué frecuencias están presentes en una señal, pero no en qué lugar en el tiempo se han producido [3]. Un mejor enfoque para analizar señales con un espectro de frecuencias dinámico es la Transformada Wavelet. Esta tiene una alta resolución tanto en el dominio de la frecuencia como en el del tiempo. Además, la Transformada Wavelet puede proporcionar una resolución de frecuencia variable, lo que significa que puede adaptarse a diferentes escalas de tiempo y frecuencia. Esto puede ser útil en la clasificación de géneros musicales, donde ciertos géneros pueden tener patrones rítmicos más rápidos o lentos que otros [6]. Por último, la Transformada Wavelet, ya que utiliza pocos datos para representar una señal puede ser útil en la clasificación de géneros musicales; donde el procesamiento de grandes cantidades de datos suele ser costoso en términos de tiempo y recursos computacionales.

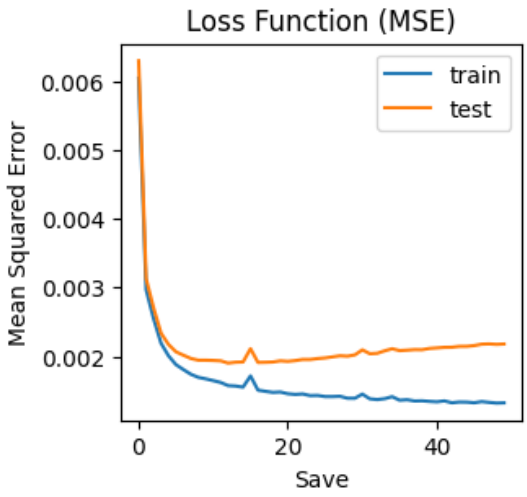
II. Propuesta

1. Autoencoder

Una de las ideas implementadas para hacer uno de los clasificadores de nuestro ensemble combina análisis de la letra con un embedding y de la música con un encoder. La idea de este modelo es concatenar los vectores resultantes de estos procesamientos y clasificar en base a esta conbinación de features.
Para construir el encoder se programó un autoencoder y se tomó el modelo hasta el bottleneck para sacar el encoder. La arquitectura del autoencoder combinó capas maxpooling y upsampling al inicio y al final respectivamente para moderar el tamaño la imagen, intercaladas con capas convolucionales y en el medio tuvo un par de capas densas para aprovechar que ya el número de dimensiones era relativamente pequeño y realizar un poco más de aprendizaje. Se tomó como función de pérdida y métrica el error cuadrático medio (*mean squared error*). La entrada del autoencoder y la salida espera-

da fueron las imágenes del feature MFCC del conjunto de entrenamiento luego de haber sido normalizadas, es decir, que en vez de estar en el rango $[0, 255]$ cada valor de la imagen de entrada, los representamos en el rango $[0, 1]$.

Anteriormente se probaron otras arquitecturas, desde algunas que no tenían capas densas hasta otras que principalmente consistían en capas densas. El problema en las arquitecturas carentes de capas densas era que sus resultados no eran lo suficientemente buenos, es decir, debido a su relativamente baja cantidad de parámetros el nivel de aprendizaje que podían lograr era inferior al que se logró luego con arquitecturas con capas densas. Por otro lado, las arquitecturas que consistían principalmente en capas densas tenían problemas como que los modelos eran muy grandes, algunos pasando de los GiB de almacenamiento y presentaban un problema para nosotros a la hora del entrenamiento. Otro problema que tienen las arquitecturas más basadas en capas densas es su tendencia al overfitting. En las prueba realizadas, las arquitecturas carentes de capas densas no presentaban este tipo de problema ya que los resultados en los conjuntos de entrenamiento, test y validación tenían poca diferencia entre ellos, sin embargo en las arquitecturas que tenían capas densas, por el gran número de parámetros si se evidencia una diferencia sustancial entre los resultados en los conjuntos de entrenamiento, y los obtenidos en los de prueba y validación. Al entrenar se realizó un *save* cada 10 epochs. Veamos los resultados sobre el número del *save* en el modelo de autoencoder presentado:



A partir del análisis de la gráfica anterior se tomó como cantidad de epochs a ejecutar la cantidad de 150, ya que se conjeturó (y luego validó) que el comportamiento del modelo en el conjunto de prueba sería muy similar al comportamiento en el conjunto de validación y en este punto es que se obtiene un mejor performance en el conjunto de prueba.

Volviendo atrás, los resultados obtenidos para cada tipo de modelo:

- los modelos que no tenían capas densas lograron primeramente un error (MSE) de 0,0025 y luego de ampliar la cantidad de dimensiones que salen del bottleneck, es decir la cantidad de dimensiones de la representación se logró 0,0021. Estos modelos tenían muy poco overfitting luego de 500 epochs.
- los modelos que presentan capas densas, por su parte, comenzaron con resultados en el conjunto de entrenamiento de hasta 0,0011 con 500 epochs, lo cual era muy bueno pero podía implicar overfitting. A medida que se redujeron la cantidad de parámetros (de 288 millones a los 2,3 millones del modelo propuesto) los resultados en el conjunto de entrenamiento fueron peores pero nunca sobrepasaron el valor de 0,0013 en 500 epochs. Luego de correr el modelo propuesto solo 150 epochs se obtuvieron los mejores resultados tanto en el conjunto de prueba como en el validación, oscilando alrededor de 0,00185, por su lado en el conjunto de entrenamiento se obtuvieron resultados alrededor de 0,0016, lo que evidencia la presencia de overfitting.

Se realizó también cross validation con Kfold dividiendo todo el conjunto de entrenamiento en 10 subconjuntos. Los resultados del cross validation coincidieron con los resultados anteriormente descrito. Este test se hizo luego de haber fijado la cantidad de epochs en 150.

II. Wavelets

Proponemos dos métodos de extracción de características usando varias formas de Transformadas Wavelet. El primero es la Transformada Wavelet Discreta (DWT) [3] y el segundo la Transformada Wavelet Compleja de Doble Árbol (DT-CWT) [4].

La DWT es un caso especial de Transformada Wavelet que proporciona una representación compacta de la señal en el tiempo y la frecuencia que se puede calcular de manera eficiente[6]. La DT-CWT es una mejora relativamente reciente a DWT; ya que para señales moduladas complejas como el audio, DWT encuentra algunas pocas deficiencias: oscilaciones, varianza de desplazamiento, aliasing y falta de direccionalidad[2].

El pipeline para ambos métodos consiste en calcular las respectivas características para cada canción del dataset. Luego con la matriz obtenida se realiza un split de 80 % - 20 % y se entrena el modelo de machine learning elegido. Ajustando los hiperparámetros utilizando Cross-Validation.

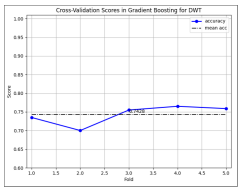
TODO

III. Tests

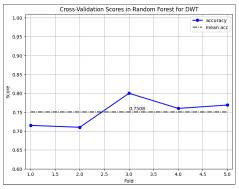
TODO

Para la implementación utilizando Transformadas Wavelet fueron probados varios modelos de machine learning tradicional como Logistic Regression, SVC, Linear SVC, Random Forest Classifier y Gradient Boosting Classifier. Como los dos últimos se comportan mejor en aproximadamente más de 10 % de precisión respecto al resto, decidimos enfocarnos en esos modelos .

Respecto a la Transformada Wavelet Discreta, Daubechies wavelet empíricamente, muestra el mejor comportamiento en muchas aplicaciones[2]. Experimentamos con distintos órdenes de Daubechies wavelet, y db12 mostró el mejor comportamiento. Los mejores resultados para DWT fueron obtenidos con Random Forest; utilizando como parámetros: *n_estimators=100*, *max_depth=13*, *bootstrap=False*. La precisión con este modelo se ubicaba alrededor de 0,77.

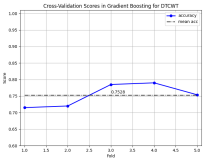


(a) *DWT Gradient Boosting*

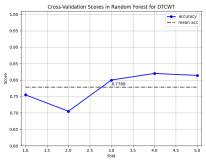


(b) *DWT Random Forest*

Figura 1: *DWT Cross-Validation*



(a) *DTCWT Gradient Boosting*



(b) *DTCWT Random Forest*

Figura 2: *DTCWT Cross-Validation*

La Transformada Wavelet Compleja de Doble Árbol, mostró los mejores resultados con 17 niveles de descomposición para extraer los coeficientes wavelet. Se comporta mejor que DWT, alcanzando más de 0,8 de precisión con Random Forest Classifier. Además de las imagenes del Cross-Vlidation también reportamos una matriz de confusión para comprobar el comportamiento en cada género. Se observa que no se desempeña de la misma forma en todos los géneros, ya que hay algunos (como el metal o el jazz) donde se observan muy buenos resultados.

IV. Recomendaciones

TODO

Referencias

[1] Kevin P. Murphy: *Machine Learning: A Probabilistic Perspective*. MIT Press, 2012.

[2] Pranav Vijaya Kumar Rao, Vishwas Nagesh Moolimani: *ECG Analysis based feature extraction using Wavelet Transform for Music Genre Classification*, 2020.

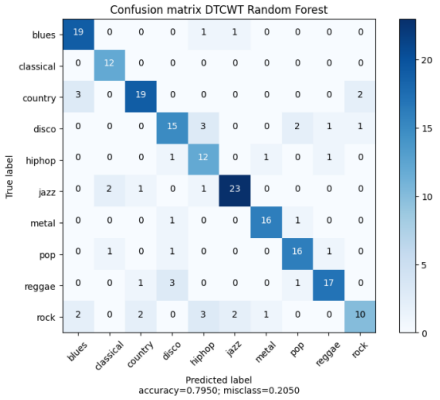


Figura 3: *DTCWT Random Forest Matriz de Confusión*

[3] Ahmet Taspinar: *A guide for using the wavelet transform in machine learning*, unpublished. [Online]. Available: http://ataspinar.com/2018/12/21/a-guide-for-using-the-wavelet-transform-in-machine-learning/

[4] Selesnick, I.W. and Baraniuk, R.G. and Kingsbury, N.C., *The dual-tree complex wavelet transform*, 2005, IEEE Signal Processing Magazine, pp. 123-151.

[5] Liliana R. Castro, Silvia M. Castro: *Wavelets y sus Aplicaciones*, ler. Congreso Argentino de Ciencias de la Computación, pp. 195-204.

[6] George Tzanetakis, Georg Essl, Perry Cook: *Automatic Musical Genre Classification Of Audio Signals*

TODO