Pregunta 1

Sin responder aún

Puntúa como 30,00

OJO: Aquellos que tengan problemas con el reproductor de audio en su pc pueden utilizar alguna de las siguientes herramientas online para jupyter notebooks: googlecolab o deepnote para ejecutar sus notebooks antes de subirlos.

Link a archivo base de resolución de tarea 2: Archivo.

En esta tarea estudiaremos la descomposición de valores singulares, o también conocida como la factorización matricial SVD.

Lo interesante de esta factorización matricial es que siempre existe para matrices de cualquier dimensión, sean cuadradas o no.

La factorización SVD tiene la siguiente forma para una matriz $A \in \mathbb{R}^{m \times n}$:

$$A = U \Sigma V^*$$

donde U es una matriz unitaria, $\Sigma = diag(\sigma_1, \sigma_2, \cdots)$ es un matriz diagonal con σ_i el i-ésimo valor singular que tradicionalmente cumplen que $\sigma_i \geq \sigma_{i+1}$, V es una matriz unitaria, y $*$ es el operador de transpuesta-conjugada.

Se recuerda que en el Anexo A de los apuntes del curso se presenta una explicación más detallada de la SVD.

El algoritmo que se estudiará es el "Análisis de Componentes Principales", o PCA por sus siglas en inglés. Se recuerda también que en el Anexo B de los apuntes del curso se presenta la relación entre la SVD y PCA.

El contexto con el cual se trabajará será con un conjunto de datos de pistas de audio de una misma canción y otro conjunto de datos de audios de otra canción, a los cuales se les aplicará una reducción de dimensionalidad y se comparará la cantidad de memoria requerida en almacenar los audios originales v/s la versión comprimida, además de la "calidad" de los audios obtenidos al "reconstruir" y que tanto disminuye el ruido en los audios.

Lamentablemente no es posible obtener buenos resultados si aplica reducción de dimensionalidad separando la pista de audio en segmentos de audio mas cortos y luego aplicar PCA, sin embargo, se puede utilizar la "Short-time Fourier Transform" (STFT) para pre-procesar la pista de audio y luego "comprimirla".

¡Los resultados son de una excelente calidad! Si es que se aplica la reducción de dimensionalidad de forma adecuada.

Breve descripción de la STFT

> La STFT (Short-time Fourier Transform) corresponde a una transformación utilizada para determinar la frecuencia sinusoidal y la fase de secciones locales de una señal que cambia a lo largo del tiempo. Para tiempos discretos (que es nuestro caso) se calcula de la siguiente manera:
> $STFT\{x[n]\}(m, w) = \sum_{n=-\infty}^{\infty} x[n]\, w[n-m]\, \exp(-j\,\omega\, n)$
>
> Donde:

- $w[n-m]$ corresponde a una función ventana (por defecto scipy usa Hann Window).
- $x[n]$ corresponde a la señal a ser transformada.
- ω corresponde a la frecuencia.
- j corresponde a la unidad imaginaria, es decir $j^2 = -1$.

Esta transformación trabaja el audio segmentándolo y aplicando la Transformada de Fourier Rápida para cada uno de los segmentos, entregando como resultado una matriz compleja. Esta explicación de STFT es bien breve ya que se utilizará scipy para calcularla (no tendrán que implementarla). Puede revisar el siguiente enlace para mayor información sobre la STFT

En esta tarea se buscará comprimir un archivo de audio *WAV* con tal de reducir levemente el ruido en la pista. Este archivo consiste en una secuencia de datos muestreados en una tasa llamada _sample rate_.

Dado que *SVD* y *PCA* son utilizados tradicionalmente para reducir la dimensionalidad de datos en forma de vectores, primero tendremos que representar el audio como un conjunto de vectores de datos.

Por esta razón es que se utilizará la STFT (Short-time Fourier Transform), y está entregará la matriz de datos reducida.

Con lo anterior en mente a continuación se detalla cómo comprimir y descomprimir señales utilizando la STFT.

Flujo de trabajo de la compresión/descompresión de la señal

El flujo que seguirá la tarea será el siguiente:

Algoritmo de Compresión:

1. Aplicar STFT al audio obteniendo M.
2. Aplicar Split a M obteniendo M_Re y M_Im.
3. Aplicar PCA a M_Re y M_Im con $m1$ y $m2$ componentes, respectivamente.

Algoritmo de Descompresión

1. Reconstruir las matrices M_Re y M_Im luego de aplicar PCA.
2. Aplicar Merge a las nuevas matrices M_Re y M_Im para obtener la matriz M modificada.
3. Aplicar ISTFT a la matriz M modificada para recuperar el formato de audio (arreglo de datos).

ETAPA 1: Obtención de PCA y errores/compresión.

Se dan 6 funciones implementadas y se pide implementar 5:

- Función 1: La primera función se denomina WAV_to_Array. Esta función retorna un int sample_rate y un NumPy array 1-D “data” con los datos de la pista de sonido cargada.
- Función 2: La segunda función se denomina play_audio. Esta función retorna un widget que permite reproducir el audio ingresado.
- Función 3: La tercera función se denomina STFT. Esta función retorna un arreglo (p, q) complejo, es decir la matriz compleja M.
- Función 4: La tercera función se denomina ISTFT. Esta función retorna un numpy un NumPy array 1-D “data” con los datos de M transformados nuevamente a una pista de sonido.
- Función 5: Split. Esta función retorna dos arreglos (p, q) reales, es decir la matriz compleja M dividida en sus componentes reales e imaginarias.
- Función 6: Merge. Esta función retorna un arreglo (p, q) complejo a partir de dos arreglos (p, q) reales, es decir la matriz compleja M formada a partir de su parte real e imaginaria.

Se deben implementar las siguientes funciones haciendo uso de las funciones anteriores según se estime conveniente:

- [15 puntos] Función 7: PCA_SVD. Esta función aplica PCA sobre M recibiendo los m componentes que se utilizarán, retornando el arreglo V de (q, m) vectores singulares, el arreglo Y de (p, m) de coeficientes singulares y μ arreglo de (q) de medias por columna.
- Para esto debe utilizar la función de SVD de scipy llamada linalg.svd. Ponga atención si va a requerir la versión reducida o full de la SVD.
- [15 puntos] Función 8: La octava función se denomina PCA_M. Esta función retorna la matriz M arreglo (p, q) reconstruida con m componentes principales.
- [10 puntos] Función 9: La novena función se denomina calc_compression_ratio. Se debe implementar una función que obtenga el costo de almacenamiento de aplicar PCA a M (la matriz de m vectores principales V, los coeficientes proyectados Y, y el vector μ para la matriz) dividido en el costo de almacenamiento del vector de datos originales (conjunto de audios) AUDIOS. Esta funciona retorna un float
- La función debe retornar la tasa de compresión obtenida del siguiente cálculo:
- $compression\_ratio = \left(1 - \frac{memoria\_comprimida}{memoria\_original}\right) * 100$
- Notar que la memoria requerida es básicamente la cantidad de coeficientes que tiene que almacenar en cada caso.

- [15 puntos] función 10: La décima función se denomina compression algorithm. Esta función retorna V_1, Y_1, μ_1 arreglos de $(q, m), (p, m)$ y (q) respectivamente para el componente real de M y V_2, Y_2, μ_2 arreglos de $(q, m), (p, m)$ y (q) respectivamente para la componente imaginaria de M. Hint: Algoritmo de Compresión.
- [15 puntos] función 11: La enésima función se denomina decompression_algorithm. Esta función retorna la nueva pista de audio comprimida "data_modified" la cual es un numpy array 1-D. Hint: Algoritmo de Descompresión.

Basta con que entregue las funciones. No responda la pregunta textualmente, recuerde que no se deben hacer prints ni pedir inputs en la tarea.

ETAPA 2: Experimentación.

Ejecute la sección de experimentación utilizando la el archivo "output/song_2/vocals.wav" para cargar las pistas de audio y responda las siguientes preguntas:

[5 puntos] ¿Es razonable usar una sola componente principal para comprimir toda la pista de audio?. Responder 1 o 0, donde 1 indica sí y 0 indica no.

[5 puntos] ¿En relación al gráfico cuántas componentes minimizan el error generado por el ruido?. Responder de acuerdo a lo mostrado en el gráfico (puede modificar los valores hasta encontrarlo).

[5 puntos] ¿Tener un índice de compresión negativo es indicativo de un mayor ahorro de memoria? (valores bajo la línea naranja en la gráfica de la izquierda). Responder 1 o 0, donde 1 indica sí y 0 indica no.

[5 puntos] Usando 50 componentes principales: ¿Se está minimizando el ruido?. Responder 1 o 0, donde 1 indica sí y 0 indica no.

Ejecute la sección de experimentación utilizando la el archivo "output/song_1/other_noise.wav" para cargar las pistas de audio el cual consiste en un archivo de audio con ruido ya integrado por lo cual debe cargarlo sin agregar ruido extra es decir cuando se haga la carga de "data_noise" realice la siguiente modificación:

WAV_to_Array(filename, False)

Con este archivo responda las siguientes preguntas:

[5 puntos] ¿Cuántos componentes permiten obtener la menor cantidad de ruido?. Hint: No se guíe por los gráficos de errores y utilice sus oídos para determinar el menor error, no existe un único valor correcto pero su valor debe estar dentro de un intervalo

[5 puntos] Para esta señal de audio: ¿La discrepancia entre el mínimo error del gráfico y el mínimo ruido encontrado se debe a que los datos utilizados han sido alterados?. Responder 1 o 0, donde 1 indica sí y 0 indica no.

Actividad previa

◄ Jupyter Notebook Tarea 0 - parte 2

Ir a...

Siguiente actividad

Jupyter Notebook Tarea 2 - parte 2 ▶