Práctica 2: Dynamic Time Warping (DTW)

Antonio José Moya Díaz

21 de noviembre de 2012

INTRODUCCIÓN.- A continuación se hará una breve presentación y explicación del código empleado para la realización de la presente práctica. Nos centraremos en explicar el uso del código así como algún que otro concepto más teórico dejando de lado mayores explicaciones relacionadas con la programación ya que el propio código se encuentra lo suficientemente comentado.

1. Ficheros

Esta práctica ha sido planteada y diseñada mediante el uso de funciones con la intención de separar y agrupar las distintas tareas de una forma conceptual, así como facilitar el reuso de código o, incluso, habilitar la escalabilidad del mismo.

Así pues se han definido las funciones "dtw", "grabar_muestra", "grabar_patrones" y "recorta_audio". Todas ellas, que se encuentran en ficheros .m homónimos como dicta Matlab, serán explicadas en la siguiente sección. Además se incluye un script denominado "demo.m" en el que se ha creado un pequeño flujo en el que se automatiza una demostración del funcionamiento del algoritmo DTW.

2. Funciones

Todas las funciones incluyen un texto de ayuda. Ejecutando "help nombrefunción" en la command window de Matlab podrá obtener información concisa sobre su funcionamiento.

grabar_patrones

funcionamiento Esta función es la encargada de grabar la base de datos de patrones o audios a reconocer. Su ejecución preguntará al usuario cuántas muestras (patrones) desea grabar. Luego pedirá al usuario que introduzca una cadena de texto con el texto que va a grabar (o cualquier otro texto que permita identificar a nivel de usuario el resultado del reconocimiento) y acto seguido demandará al usuario que pronuncie el patrón a grabar. Finalmente, iterará estos dos pasos tantas veces como número de patrones se haya introducido al principio.

entrada La función puede funcionar por defecto sin ningún parámetro de entrada. Opcionalmente se permite la definición de los parámetros del grabador, como son la frecuencia de muestreo y el número de bits por muestra. Por defecto éstos son, respectivamente, 8kHz y 16 bits. El código no está preparado para trabajar con más de un canal, por lo que todas las grabaciones están limitadas a un audio "mono".

salida La función devuelve un elemento de tipo celda de n filas y 2 columnas. En la primera columna contiene una cadena de texto con el texto que contiene la muestra de audio. Muestra que se encuentra encuentra en la misma fila en la columna 2. La celda contendrá una fila por cada muestra grabada.

grabar_muestra

funcionamiento Esta función, muy similar a la anterior, se encarga de la grabación de la muestra a reconocer. Ha sido pensada como una función aparte de la anterior ya que no necesita iterar.

entrada En cuanto a los parámetros de entrada, esta función funciona exactamente igual que la función "grabar_patrones".

salida La función devuelve una señal de audio preparada para su posterior uso en la función de reconocimiento DTW.

dtw

funcionamiento Es la función que implementa el algoritmo DTW. Esta función toma una señal patrón y una señal incógnita, y les aplica un filtrado de preénfasis en primer lugar. Luego las segmenta en tramas y calcula las distancias euclídeas según la potencia del espectro de Fourier. Con esas distancias crea, en primer lugar, un matriz de distancias. Una vez creada ésta, genera una segunda matriz de distancias acumuladas según marca el algoritmo DTW para finalmente quedarse con el elemento último de la matriz que será la distancia que nos interesa.

Notar una diferencia con respecto al algoritmo planteado en clase. Éste recorría la matriz comenzando por la esquina inferior izquierda y acababa en la esquina superior derecha. Por evitar trabajar con índices decrecientes, y ya que no altera el algoritmo, nuestra matriz de distancias acumuladas es recorrida desde la esquina superior izquierda hasta la inferior derecha.

entrada Esta función recibe como entrada las dos señales, el patrón y la incógnita.

salida La función devuelve un único número que representa la distancia total acumulada por el algoritmo.

recorta_audio

funcionamiento La tarea de esta función es dada una muestra de audio de longitud arbitraria, recortarla conservando únicamente la zona que contiene la información de la señal con un margen temporal por delante y por detrás para garantizar que no se corta trozo de señal que contenga información.

Para eso se implementa un muy sencillo reconocimiento de actividad vocal, y sobre los índices marcados por éste se añade el relleno necesario. Para evitar problemas que se comentarán en la sección 3 de este documento, para que el recorte sea efectivo se fuerza a que haya un espacio sin actividad vocal tanto por delante como por detrás. En caso contrario, devolverá un vector vacío a modo de señal de error.

entrada La función recibe como entrada la propia señal, la frecuencia de muestreo y un padding o relleno, expresado en segundos. Adicionalmente puede recibir un parámetro opcional para representar gráficamente la zona del corte y el corte ya realizado.

salida La función devuelve una señal de audio que contiene la señal de actividad vocal con su debido relleno, o un vector vacío como indicación de que la señal de entrada no es válida.

3. Observaciones

Creo necesario el comentar algunos detalles que se han observado a lo largo del desarrollo del código.

En primer lugar puede darse un problema derivado del detector de actividad vocal (VAD). La inclusión del VAD está justificada desde 2 puntos de vista. En primer lugar tener una muestra pequeña y acotada de lo que queremos reconocer y, además, su implementación nos permitirá descartar algunos casos que falsearían el reconocimiento.

El VAD no es más que un umbral que buscará el primer y último punto en el que la señal superará un valor fijado a un tercio del valor máximo de la señal. A partir de dichos puntos se añadirá el relleno y realizará el recorte. Esto obliga a que la muestra vocal se encuentre, aproximadamente, en el centro de la grabación. Esto se ha hecho así por la siguiente razón. Si uno de los extremos de la señal careciese de espacio de relleno -ésto es que la señal contuviese valores superiores al umbral, o lo que es lo mismo, información vocal- en las muestras primera o última de la señal no podríamos garantizar que todo el audio a reconocer haya sido grabado.

Lógicamente no se puede garantizar el reconocimiento si tanto el patrón como las incógnitas no contienen la información vocal necesaria, por eso hemos forzado la condición anterior de que haya espacio suficiente para relleno tanto antes como después de los valores detectados por el VAD.

A colación de lo anterior, aunque su influencia es menor, un ruido impulsivo de gran potencia, durante la grabación podría hacer que la señal recortada contuviese más audio del estrictamente necesario lo que a posteriori hará que baje bastante la tasa de aciertos.

También se ha detener en cuenta que el algoritmo, si bien aceptablemente robusto frente a ruido, es altamente sensible a cambios de entonación o diferencias en la pronunciación. Algo tan simple como intentar reconocer por la mañana temprano, recién levantado, usando unos patrones grabados la noche anterior, descenderá la tasa de aciertos hasta niveles realmente bajos.

4. Resultados

En cuanto a los resultados se pensó en incluir una estadística de los mismos pero se descartó debido a la cantidad de intentos que se deberían hacer para mostrar una estadística medianamente fiable.

Hay que decir que con todos los comentarios hechos hasta el momento es fácil intuir los resultados. Éstos son en general bastante buenos siempre que las muestras de audio hayan sido grabadas correctamente y dentro de las limitaciones anteriormente expresadas. Sin embargo el algoritmo falla con facilidad con cambios tonales de los distintos audios, y cuando ésto ocurre suelen ser los patrones con menor número de muestras los que acaban resultado con distancias más pequeñas.