Práctica 2: Reducción de imágenes usando familias A y B y la composición max-min

En el Tema 2 de las clases teóricas hemos estudiado la composición max-min entre dos conjuntos difusos para obtener una relación de la forma

$$R(x,y) = \bigvee A(x) \wedge B(y)$$

En esta práctica vamos a basarnos en esta composición para reducir la cantidad de datos necesarios para almacenar una imagen. En primer lugar, vamos a considerar que una imagen no es más que una relación difusa *R* en la que cada píxel (elemento de la matriz) tiene una intensidad, que es un valor en el intervalo [0,1]. Sin embargo, obtener una matriz parecida a la original a partir de dos conjuntos difusos A y B es realmente complicado. Por ello, vamos a considerar que podemos obtener la imagen como

$$\tilde{R}(x,y) = \bigvee_{i=1}^{c} (A_i(x) \wedge B_i(y))$$

donde *c* (también llamado Schein rank de la relación *R*) es un numero natural que satisface $c \geq 2$. Es evidente que cuanto mayor es el valor de c, mejor será la aproximación de R.

El **objetivo** de esta práctica es: calcular una familia de *c* conjuntos difusos *A* y *c* conjuntos difusos *B* tales que al componerlos utilizando la expresión anterior obtengamos una aproximación de la imagen original *R*. Es evidente que si el valor de *c* no es muy elevado, estaremos almacenando menos información en nuestros conjuntos *A* y *B* que en la imagen original.

¿Cómo vamos a ser capaces de calcular estos conjuntos *A* y *B*? En primer lugar, vamos a partir de valores aleatorios. Para obtener los valores de *A* y *B* óptimos vamos a utilizar la técnica del gradiente. Para ello, consideramos que el problema de descomponer la relación *R* en conjuntos *A* y *B* puede ser visto como un problema de optimización que trate de minimizar la siguiente función de coste

$$Q = \sum_{(x,y) \in X \times Y} (R(x,y) - \tilde{R}(x,y))^2$$

Observa que, si al componer los conjuntos *A* y *B* obtenemos la misma imagen de la que partimos, entonces $Q = 0$ y hemos encontrado el mínimo de la función. El proceso de encontrar estos valores de *A* y *B* va a ser iterativo. Denotaremos por *iter* el nº de

iteración que estamos tratando actualmente. Así, la función de coste en cada iteración puede reescribirse como

$$Q_{(iter)} = \sum_{(x,y)\in X\times Y} \left(R(x,y) - \bigvee_{i=1}^{c} A_i^{(iter)}(x) \wedge B_i^{(iter)}(y) \right)$$

Para actualizar los valores de A y B utilizamos las siguientes fórmulas:

$$A_i^{(\text{iter}+1)} = A_i^{(\text{iter})} - \alpha \frac{\partial Q_{\text{iter}}}{\partial A_i^{(\text{iter})}(x)}$$

$$B_i^{(\text{iter}+1)} = B_i^{(\text{iter})} - \alpha \frac{\partial Q_{\text{iter}}}{\partial B_i^{(\text{iter})}(y)}$$

donde las derivadas de *Q* respecto a *A* y a *B* las calculamos como

$$\frac{\partial Q}{\partial A_l(x')} = -2 \sum_{y\in Y} \left(R(x',y) - \bigvee_{i=1}^{c} (A_i(x') \wedge B_i(y)) \right)$$

$$\varphi \left((A_l(x') \wedge B_l(y), \bigvee_{i=1, i\neq l}^{c} (A_i(x') \wedge B_i(y)) \right)$$

$$\psi(A_l(x'),\ B_l(y))$$

$$\frac{\partial Q}{\partial B_l(y')} = -2 \sum_{x \in X} \left(R(x, y') - \bigvee_{i=1}^{c} (A_i(x) \wedge B_i(y')) \right)$$
$$\varphi \left(A_l(x) \wedge B_l(y'), \bigvee_{i=1, i \neq l}^{c} A_i(x) \wedge B_i(y') \right)$$
$$\psi \left(B_l(y'), A_l(x) \right)$$

siendo

$$\varphi(a, b) = \begin{cases} 1, & \text{if } a \geq b, \\ 0, & \text{otherwise,} \end{cases}$$

$$\psi(a, b) = \begin{cases} 1, & \text{if } a \leq b, \\ 0, & \text{otherwise,} \end{cases}$$

La última cuestión que hay que tener en cuenta es cuándo debemos terminar nuestra ejecución. Al ser un proceso iterativo, necesitamos una condición de parada. Vamos a considerar que hemos llegado al final de nuestro proceso cuando en dos iteraciones consecutivas, la diferencia entre la función de coste Q en cada una de ellas sea lo suficientemente pequeña, es decir,

$$Q^{(\text{iter}+1)} - Q^{(\text{iter})} < \varepsilon,$$

Crea un programa en Matlab que reciba como entrada una imagen (de cualquier dimensión), un valor de c y un valor de ε y devuelva las familias de conjuntos A y B y la reconstrucción de la imagen R.
Ten en cuenta las siguientes consideraciones:

- $c \geq 2$
- A_i son conjuntos difusos de tantos elementos como filas tiene la matriz de entrada
- B_i son conjuntos difusos de tantos elementos como columnas tiene la matriz de entrada
- Un buen valor de α es 0.005.

- Los elementos de la imagen original una vez leída en Matlab son de tipo *uint8* (entero sin signo en el intervalo [0,255]). Debes convertirlos en elementos del tipo *double* en el intervalo [0,1]. Ver apéndice.
- Utiliza tus conocimientos de Matlab y de cómo trabajar con matrices para obtener un código lo más óptimo posible (se valorará en la puntuación de la práctica).

Deberás entregar en un archivo .zip el programa o programas de Matlab utilizados y un documento donde expliques algunos de los siguientes aspectos:

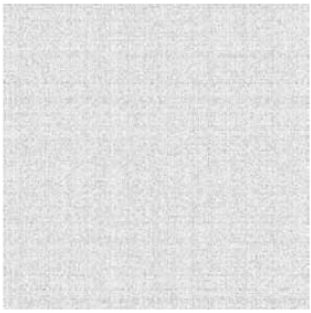
- Tiempo de ejecución del algoritmo
- Cómo afecta el parámetro c a la calidad del resultado
- Ejemplos con diferentes imágenes
- ¿Qué ocurre si cambiamos la expresión de la derivada de Q por la siguiente expresión? Analiza el porqué de este cambio y en qué puede mejorar al algoritmo su utilización.

$$\frac{\partial}{\partial A_l^{(iter)}(x')} = -2\sum_{y\in Y}\left(R(x',y) - \bigvee_{i=1}^{c}(A_i(x'),B_i(y))\right) \phi(A_l(x'),\tilde{R}(x',y))$$

$$\frac{\partial}{\partial B_l^{(iter)}(y')} = -2\sum_{x\in X}\left(R(x,y') - \bigvee_{i=1}^{c}(A_i(x),B_i(y'))\right) \phi(B_l(y'),\tilde{R}(x,y'))$$

$$\phi(a,b) = \begin{cases} 1 & \text{if } a = b \\ 0 & \text{otherwise} \end{cases}$$

Imagen original



Reconstrucción a partir de las familias A y B creadas aleatoriamente

Solución con c = 50

Solución con c = 100

Apéndice: trabajar con imágenes en Matlab

En Matlab una imagen en escala de grises está representada por medio de una matriz bidimensional, *R*, de n x m elementos, donde n (filas) representa el número de píxeles de alto y m (columnas) el número de píxeles de ancho. El elemento *R(1,1)* corresponde con el píxel de la esquina superior izquierda de la imagen. Cada elemento de la matriz tiene un valor que representa la intensidad (en niveles de gris) del píxel correspondiente. Cuando leemos una imagen en Matlab, cada elemento de la matriz es una variable de tipo *uint8*, es decir, un entero sin signo de 8 bits (perteneciente al rango [0,255]). Sin embargo, podemos modificar la matriz para poder operar con los píxeles como si fuesen valores reales.

A continuación encontrarás las funciones básicas para el tratamiento de imagen. Recuerda que siempre puedes utilizar *help* de Matlab.

- Leer una imagen: *R = imread('nombre_de_la_imagen');*
  Devuelve en *R* la matriz correspondiente a la imagen. (Ej: *R = imread('circulo.jpg');*)
- Tamaño de una imagen: *[filas, columnas] = size(R);*
- Transformar los valores de los píxeles a formato double: *R = double(R);*
- Transformar los valores de los píxeles a formato uint8: *R = uint8(R);*
- Almacenar una imagen: *imwrite(R, 'nombre_de_la_imagen');*
  Escribe la matriz R en la carpeta en la que estés trabajando.
  (Ej: *imwrite(R,'circulo2.jpg');* Recuerda que para poder visualizar bien la imagen *R* debe estar en formato uint8.
- Mostrar una imagen: *imshow(R)*. Recuerda que para poder visualizar bien la imagen *R* debe estar en formato uint8 o bien ser double con valores entre [0,1].