PONTIFICIA UNIVERSIDAD CATÓLICA DE CHILE

DEPARTAMENTO DE INGENIERÍA INDUSTRIAL Y SISTEMAS

ICS2121 - MÉTODOS DE OPTIMIZACIÓN

Eliminación de ruido en imágenes

Integrantes
VICENTE ARRIAGADA - 16640799
WALDO GONZÁLEZ - 16637534
MOISÉS SAAVEDRA - 16636813
JOAQUÍN TERREROS - 16205642

19 de noviembre de 2018

Introducción al problema propuesto

El problema de la eliminación del ruido en las imágenes deriva del interés actual de poder obtener la mejor calidad al momento de tomar una fotografía. Esto, a pesar de que la tecnología ha producido mejores aparatos fotográficos, de mejor calidad y resolución, pero a´un sigue existiendo situaciones externas que afectan el producto final al momento de capturar una imagen (la estabilidad del fotógrafo, la luminosidad del espacio, el movimiento del objeto que desea ser capturado, entre otros).

Es por esto que se investiga sobre una forma de poder *limpiar* las imágenes obtenidas, principalmente a través de dos caminos:

1. Suponiendo la tenencia de un diccionario que nos permita descifrar el significado de la imagen perturbada, de tal forma de que, al momento de capturarla, venga en una forma codificada y sea la persona la que implemente una metodología que permite ir variándola para obtener un *significado* (de ahí el uso de un diccionario) deseable.

2. Por otra parte, se buscará que el mismo diccionario, anteriormente mencionado, se vaya expandiendo y mejorando en base a las mejores soluciones buscadas.

Siempre manteniendo en mente, para ambos casos, que se intentará hacerlo al menor esfuerzo, es decir, utilizando la menor cantidad de *búsqueda* en el diccionario (Florent, *atal*, 2011).

Método para abordar el problema

Un método muy conocido y utilizado para solucionar problemas surgidos en el procesamiento de señales e imágenes es ISTA (Iterative Shrinkage-Thresholding Algorithm), el cual, siendo una extensión del algoritmo de gradiente clásico, es capaz de resolver problemas a gran escala incluso teniendo matrices *densas*. Sin embargo, este método, a pesar de ser atractivo por su simplicidad, es conocido por converger lentamente a una solución. De aquí nace uno nuevo: FISTA (Fast Iterative Shrinkage-Thresholding Algorithm), el cual preserva la simplicidad computacional de ISTA pero con una tasa de convergencia que ha demostrado ser significativamente mejor, tanto en teoría como en práctica. Dentro del procesamiento de imágenes surgen problemas lineales inversos, los cuales llevan a estudiar un sistema lineal discreto de la forma:

$$Ax = b + w$$

donde $A \in R^{mxn}$ y $b \in R^m$ son conocidos, w es un vector de ruido (o perturbación) desconocido y x es la imagen *verdadera* y desconocida que se debe estimar. En problemas de imágenes borrosas, por ejemplo, b representa la imagen borrosa y x es la imagen verdadera desconocida, cuyo tamaño se supone que es el mismo que el de b ($m = n$). En estas aplicaciones la matriz A describe el *operador de desenfoque*, siendo el problema de estimar un x, dada la imagen borrosa y ruidosa observada b, uno denominado problema de desenfoque de la imagen (Beck, Amir y Teboulle, Marc, 2009).

En este sentido introduciremos el concepto de diccionario $D \in R^{mxk}$ que corresponde a una matriz que formará nuestro vector $x \in R^m$ como una combinación lineal de pesos de un vector $\alpha \in R^k$.

Una vez hecha explicada a grandes rasgos el problema de optimización generamos un modelo:

$$\min_{S} \quad \frac{1}{n_s}\|s_i - Wb_i\|_2^2 \tag{1}$$

$$\min_{S} \quad \frac{1}{n_s}\|s_i - Wb_i - D_s\alpha^*(b_i, D_b)\|_2^2 \tag{2}$$

$$\alpha^*(b_i, D_b) = \underset{\alpha \in R^k}{\operatorname{argmin}} \quad \|b_i - D_b\alpha\|_2^2 + \lambda\|\alpha\|_1 \tag{3}$$

En este contexto definiremos el vector b como los componentes borrosos de la imagen, s como el vector de componentes nítidos, W como la matriz de coef que ponderan los pixeles borrosos y n_s es el tamaño del vector s. Entonces en el problema de optimización (1) simplemente buscamos los componentes del vector s tal que se aproximen de mejor manera a la imagen borrosa ponderada por los coeficientes de la matriz W, como es de esperar esto puede ser resuelto mediante un modelo de FISTA ya que corresponde a un problema de LASSO.

Otra forma de plantear el problema es mediante diccionarios, donde D_s es el diccionario correspondiente al vector s, D_b al vetor b y α es el ponderador para multiplicar al diccionario D_s y generar una combinación lineal de valores de tal forma que $s \approx D_s\alpha$. De aquí dicho problema (2) requiere buscar los pesos óptimos α de tal forma que minimicen su parecido o similitud al crear la combinación lineal entre el vector b, es decir buscar el mejor ajuste para generar la diferencia más pequeña de $b - D_b\alpha$, lo anterior expresado en (3). Como es de esperar, tanto (2) y (3) se pueden formular como problemas de Lasso donde (3) es l_1.

Procedimiento inicial para abordar el tema

En esta sección intentaremos abordar por primera vez de forma analítica el problema planteado anteriormente, para así obtener una base de lo que podría ser el sistema computacional al aplicarlo.
La metodología que utilizaremos es generar un pseudo-código que nos permita entender paso a paso los resultados obtenidos.
Para poder trabajar el código, asumiremos que los siguientes problemas son equivalentes (y de hecho describiremos el porqué):

$$\alpha^*(b_i, D_b) = \underset{\alpha \in R^k}{\operatorname{argmin}} \quad \|b_i - D_b\alpha\|_2^2 + \lambda\|\alpha\|_1 \tag{4}$$

y

$$x^* = \underset{x}{\operatorname{argmin}} \quad \|Ax - b\|^2 + \lambda\|x\|_1 \tag{5}$$

Puesto que:

1. La diferencia de normas, al estar al cuadrado, da lo mismo si es de la forma $''D\alpha - b''o''b - D\alpha''$. Así mismo sucede con el problema propuesto en el enunciado de la entrega, pues se puede considerar: $Ax - b''o''b - Ax$.
2. En la funcion objetivo a la que llegamos en la entrega anterior, al compararla con la propuesta para esta entrega, lamba son correspondientes; asÃ mismo seria con la norma 1 y con la norma 2 (que al no incluirse en el numero 2 de manera explicita se asume).

Por lo tanto, para definir el pseudo-código que se utilizarÃ¡ es el problema del enunciado de esta entrega:

$$x^* = \underset{x}{\text{argmin}} \quad \|Ax - b\|^2 + \lambda\|x\|_1 \tag{6}$$

donde los parámetros del problema son:

A: Diccionario correspondiente al vector b

x: Parámetro que estamos estimando

λ: Parámetro de penalización

b: Vector que contiene la información de la imagen.

Y el pseudo-código vendría siendo:

0 Se generan las variables p_y: vector de producto interno vectorial entre b y A; $''p''$: largo del vector producto interno entre b y A; p_x: vector de producto interno matricial entre A_i y A_j; $'X'$ que inicialmente tendrá valor 0; *Grad*: Gradiente del vector; dif_{Xmax}: La diferencia entre los valores máximos del vector X; $'stop'$: Criterio de parada (como valor númerico)

1 Con lo anterior definido vamos a hacer p recorridos (conocido como for i in range), donde cada recorrido será el recorrido $'j'$ (primero 'j' igual a 1, luego a 2, y así sucesivamente) y por cada uno de estos recorridos definiremos: âzâ como la diferencia entre la componente j-ésima de p_y menos la componente j-ésima de grad y finalmente le sumamos la componente $'j'$ de X. Además, definiremos x_{tmp} que será la diferencia entre el máximo de (0,z - λ) y el máximo de (0,-z-λ).

También calcularemos $difX$ como x_{temp} menos la componente j-ésima de X y $difabsx$ que corresponderá al valor absoluto de $difX$.

2 Si $difabsx$ es mayor a cero: nuestra componente 'j' en X tomará el valor de x_tmp, nuestro nuevo grad será el grad inicial más la componente 'j' de p_x por d_x y nuestro nuevo $difXMax$ será el mayor entre $difXMax$ y $difabsx$.
Vale decir que nuestra variable $difXMax$ corresponde a la diferencia entre nuestro X que buscamos y el que se tiene como referencia, por lo cual esa es la variable que queremos hacer lo mÃ¡s chica posible.

3 Lo anterior se realizará para cada componente y terminará siempre y cuando $difXMax \geq 'stop'$ de otra forma, se vuelve al paso [1] y se sigue con las iteraciones.

Nota: el pseudo-código anterior es una adaptación de una forma regular para la resolución de problemas *Fast Lasso* que se encuentra publicado en: `https://journals.plos.org/ploscompbiol/article/file?id=10.1371/journal.pcbi.1004755.s001&type=supplementary`

Finalmente, se puede deducir que $x^*_{MC} = A^T b$ corresponde a una aproximación de la solución óptima del problema (o una traducción del problema original) ya que se estarían *interpretando* los valores de la imagen con ruido (b) a través del diccionario (A) que planteamos para el problema. Lo anterior también se puede mirar bajo otro punto de vista. Si tomamos la función objetivo y reemplazamos x^*_{MC} en esta obtenemos lo siguiente:

$$f(x^*) = \|Ax^*_{MC} - b\|^2 + \lambda\|x^*_{MC}\|_1$$

Como sabemos que $x^*_{MC} = A^{-1}b$ y que A es ortogonal tal que $AA^T = I$ tenemos lo que sigue:

$$f(x^*) = \|A(A^T b) - b\|^2 + \lambda\|A^T b\|_1$$

$$f(x^*) = \|A(A^T b) - b\|^2 + \lambda\|A^T b\|_1$$

$$f(x^*) = \|Ib - b\|^2 + \lambda\|A^T b\|_1$$

$$f(x^*) = \lambda\|A^T b\|_1$$

Lo anterior corresponde a la solución factible del problema que se enfoca principalmente en la disminución del error más que en la significancia del modelo dandolo más énfasis a la restriccíón *l1* del problema del LASSO que estamos analizando.

Análisis y resultados

Al momento de empezar el análisis, lo que se busca es aproximar mediante operaciones de matrices la operación de convolución. En el dominio de Fourier de las frecuencias es donde se aplican los filtros a las señales, acotando el espectro y obteniendo solo las frecuencias que se deseen en cada caso. Como el cálculo de la convolución para matrices de pixeles es un proceso costoso, se puede asumir que para un kernel del filtrado *pequeño*, en comparación con la imagen, la convolcuión de dicho kernel se resumen a una simple operación de vector con matriz. Recordemos que la convolución en el espacio corresponde a la multiplicación en el dominio de la frecuencia, pero el uso de la transformada discreta de Fourier (DFT) no es óptima para este caso por el simple hecho de la dificultad de conseguir la instancia de la matriz de bases para casos de matrices de pixeles de dimensiones grandes. Por lo último, se opta trabajar en el espacio aproximando la convolución como un operador de multiplicación matricial.

Para comenzar la aplicación de la teoría de forma computacional, se tuvo que trabajar cada imagen como una matriz de pixeles. Para acotar el espacio de trabajo se normalizó cada pixel en formato de blanco y negro, es decir, cada pixel toma valores entre 0 y 1 incluidos estos.

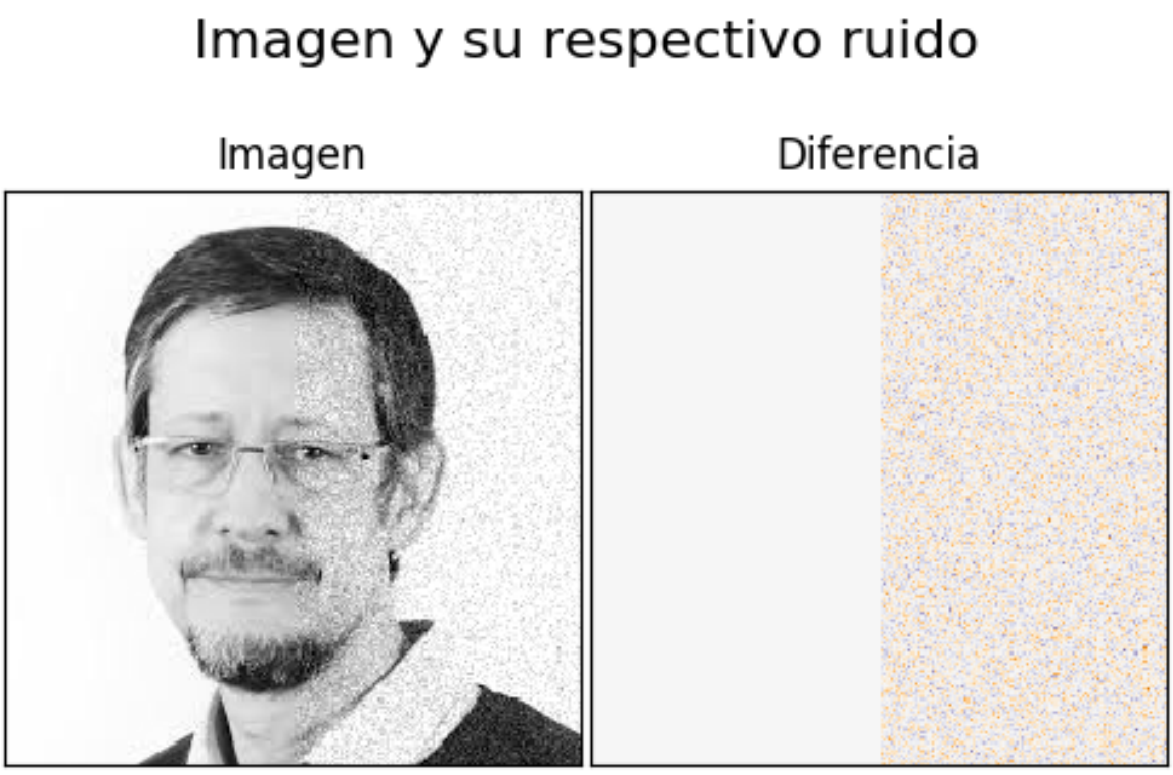
En una primera instancia se ocupó el código adjunto **Proyecto(FALLA).py** con el fin de rsolver el problema de LASSO involucrado. Se asumió que la matriz de pixeles conocida es una imagen con desenfoque (*blurry*) en la que se aplica un filtro gaussiano. Teniendo los datos listos se aplica el método de FISTA. El problema que se obtuvo fue la mala aproximación de los valores, esto se dió porque la función implementada **FISTA** es una aproximación simple y no profesional del método. Por lo que el kernel generado por el filro gaussiano no pudo ser analizado correctamente.

Ante tal problemática se buscaron formas alternativas del calculo de esta como por ejemplo la convolución mediante el método de Toeplitz. Para esto se ocupó un código de Python extraído de GitHub (referencias dentro del archivo **Convolucion.py**), lo anterior con el fin de calcular la aplicación del filtro en el dominio del espacio. El problema que surgió es que solo es un método de simulación de convolución lineal y en el caso de la convolución de imágenes este corresponde una convolución cíclica.

Finalmente, después de una ardua investigación por parte del grupo, se llega a **sklearn** que es un módulo de Python que tiene implementado una modalidad de Dictionary Learning. La gracia detrás de este módulo es que uno debe implementar los pasos necesarios y adecuar los paquetes y funciones que presenta según el problema que deseamos abordar. Para el caso de procesamiento de imágenes se ocupa y modifica un código que es presenta en una documentación del módulo (link dentro del script) que queda plasmado en el archivo **Proyecto(FUNCIONA).py**.

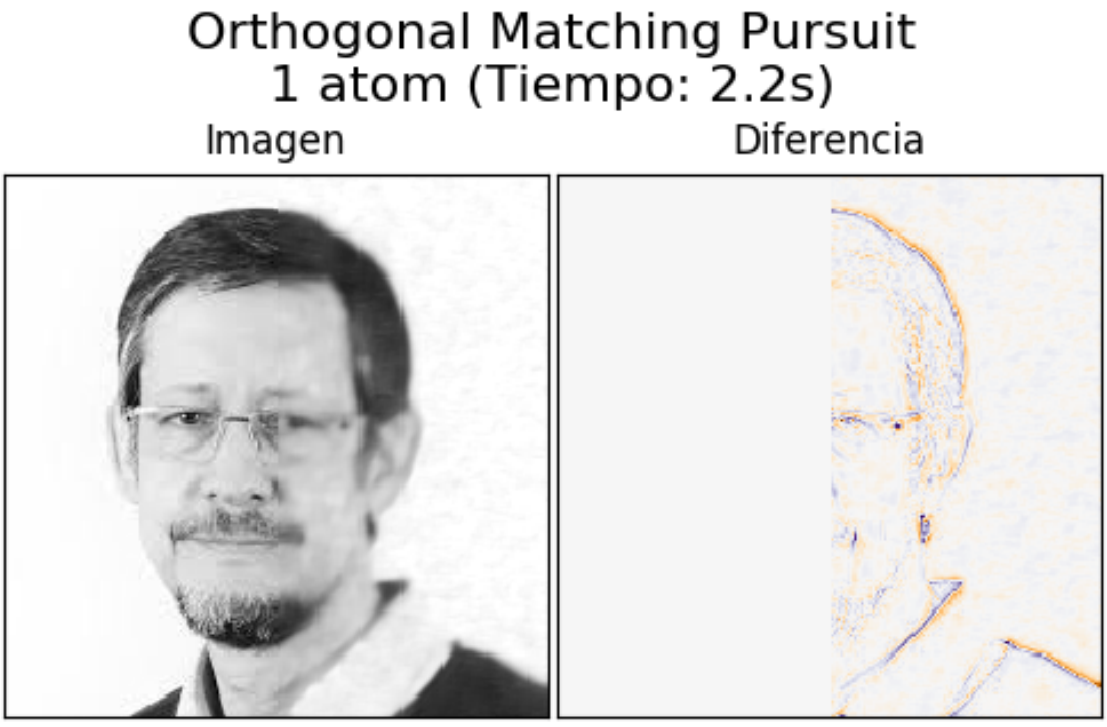
El análisis que se realiza es simple. Primero se ocupa una imagen de 256x256 pixeles y es transformada en matriz. Acto seguido, se le aplica un ruido aleatorio uniforme solamente a una mitad de la imagen con el fin de analizar como queda después del filtro de ruido. Luego se generan las instancias de diccionario según las dimensiones que el usuario desee, para nuestro caso se ocupan 100 instancias de (7x7). Una vez creadas se analizan 4 tipos de métodos distintos donde **Thresholding** es el más cercano al método de FISTA que deseabamos ocupar con un $\lambda = 0{,}1$. Finalmente, se aplica los métodos y se analizan los resultado mostrando en pantalla la imagen con una mitad correspondiente a la original, la otra mitad que es la reconstrucción de la imagen y finalmente todo el rudio de pixeles eliminado.

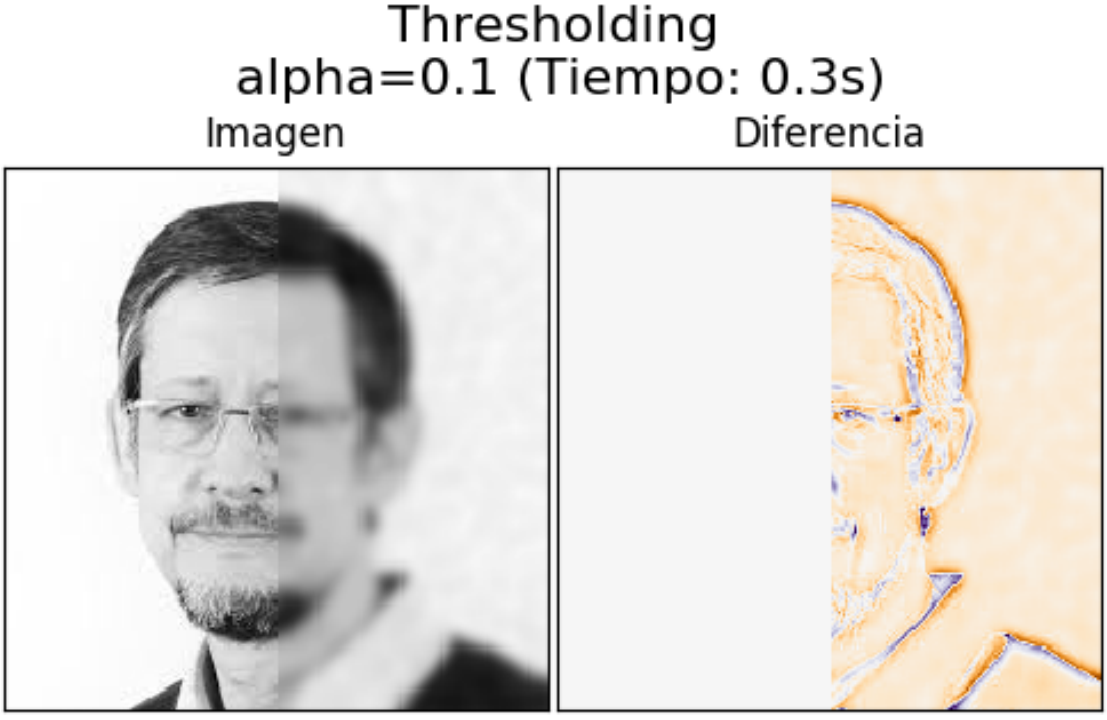
La imagen a ocupar corresponde a un **.png** que es la foto del profesor Jorge Vera:

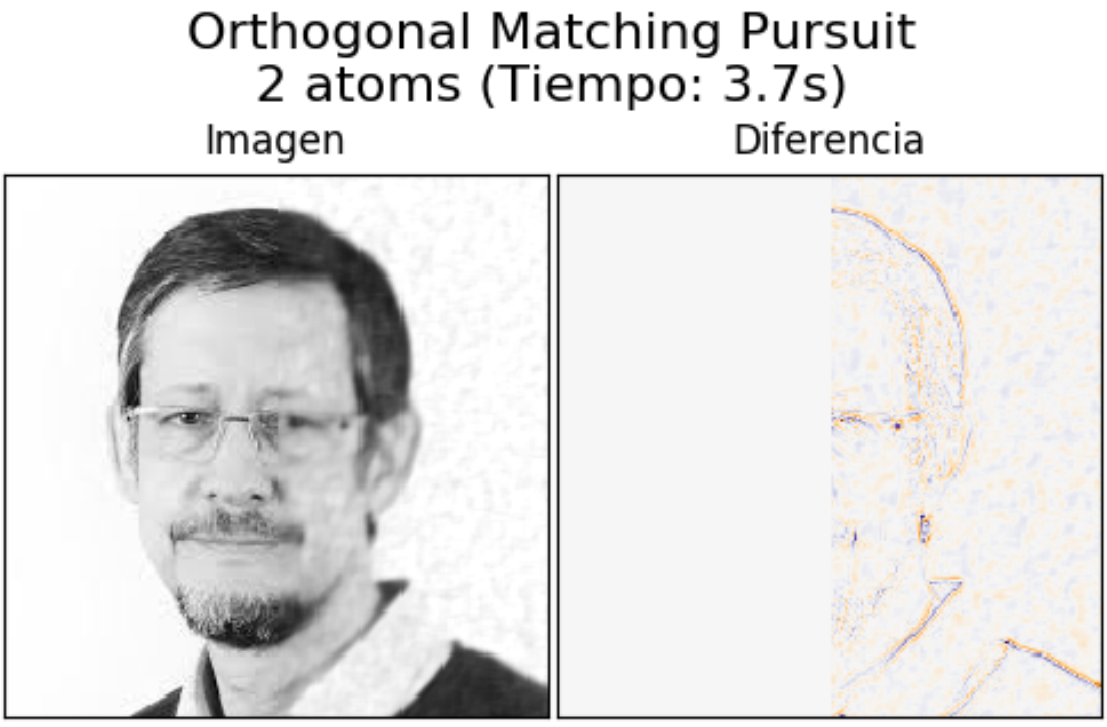


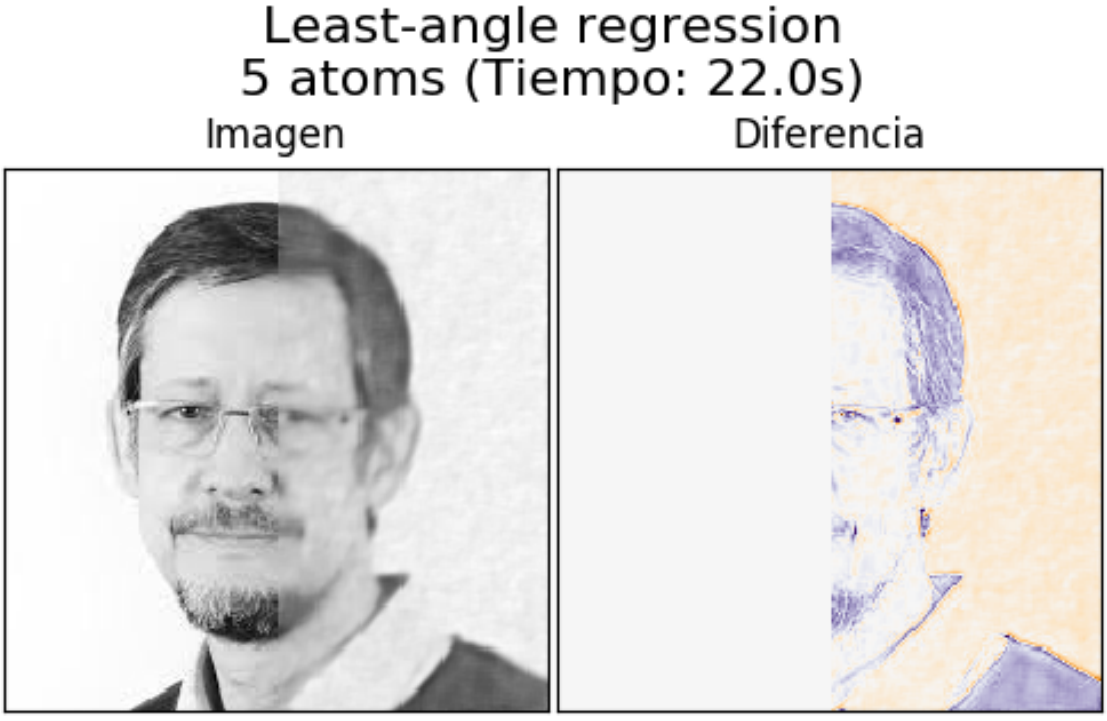
Los diccionarios de la imagen quedan como:

Y los resultados de cada método corresponden a:


Orthogonal Matching Pursuit
1 atom (Tiempo: 2.2s)
Imagen
Diferencia


Thresholding
alpha=0.1 (Tiempo: 0.3s)
Imagen
Diferencia


Orthogonal Matching Pursuit
2 atoms (Tiempo: 3.7s)
Imagen
Diferencia


Least-angle regression
5 atoms (Tiempo: 22.0s)
Imagen
Diferencia

Conclusiones

Podemos notar que a partir de los resultados el método **Thresholding** es el más efectivo en cuanto a tiempo, pero no es el mejor en reconstrucción porque si bien elimina el problema de ruido sobre una imagen hay que derivar a un segundo problema de eliminación de desenfoque (*blurry*).

Por otra parte los otros métodos (desconocidos por nosotros, pero aconsejados por la documentación) tienen tiempos de resolución bastante aceptables menos **Least-angle regresion**. La gran ventaja es que es la que mejor reconstruye si tomamos como referencia la eliminación de ruido sin generar desenfoque, pero a su vez la que más tiempo tarda.

Según las dos afirmaciones anteriores, podemos cerrar esta parte pensando que según el contexto del problema podemos ocupar cada método de resolución, ya que se genera un *trade-off* entre tiempo y error en la obtención de la solución, obviamente para contextos de tiempo como puede ser una resonancia magnética nos conviene ocupar **Thresholding**, mientras que en el filtrado de imágenes en un estudio de animaci'on puede convenir **Least-angle regression** u **Orthogonal Matching Pursuit**.

Proyecciones del problema

En primer lugar, podemos destacar la relevancia que tiene el desarrollar una metodología simple y con pocas exigencias matemáticas (en cuanto a su característica de lineal y convexa) para poder dar soluciones a este tipo de situaciones, que, a pesar del avance tecnológico en los distintos artefactos utilizados para tomar fotografías, se pueden ver afectados por situaciones cotidianas e inesperadas que produzcan que esta labor no sea totalmente completada. Sobretodo si se trata de capturas de gran relevancia, como lo son las microscópicas en la biología o la macroscópica en la astronomía.

Por otra parte, podemos concluir que el método expuesto es válido principalmente para el desarrollo de la técnica de desenfoque no digital y desenfoque *no ciego*, pero se espera que exista un desarrollo matemático y computacional más avanzado en otras áreas, como la eliminación de imágenes borrosas e incluso totalmente ciegas (imágenes sin distinción), para poder ampliar las facultades del diccionario expuesto y generar un pseudo inteligencia artificial que permita modificar las fotografías de acuerdo al contexto.

Así mismo, podemos analizar que el criterio de parada de nuestro pseudo código tiene una forma muy básica ya que solo depende de nuestra percepción sobre lo que esperamos obtener (no de un criterio objetivo uno estándar). Por lo que se podría trabajar más la metodología para obtener un formato general de aplicación a estos tipos de problemas.

Bibliografía

- Beck, Amir y Teboulle, Marc. A Fast Iterative Shrinkage-Thresholding Algorithm for Linear Inverse Problems. Society for Industrial and Applied Mathematics (2009). DOI: 10.1137/080716542. [Disponible en:
  `https://people.rennes.inria.fr/Cedric.Herzet/Cedric.Herzet/Sparse_Seminar/Entrees/2012/11/12_A_Fast_Iterative_Shrinkage-Thresholding_Algorithmfor_Linear_Inverse_Problems_(A._Beck,_M._Teboulle)_files/Breck_2009.pdf`

- Florent Couzinie-Devy, Julien Mairal, Francis Bach, Jean Ponce. Dictionary Learning for Deblurring and Digital Zoom. [Technical Report] 2011.
- https://scikit-learn.org/stable/auto_examples/decomposition/plot_image_denoising.html#sphx-glr-auto-examples-decomposition-plot-image-denoising-py
- https://github.com/alisaaalehi/convolution_as_multiplication/blob/master/ConvAsMulExplained.pdf