Experiência sobre Tomografia Computadorizada

Vítor H. Nascimento
Carlos A. Prete Jr.

27 de setembro de 2020

1. Neste exercício, vamos comparar trabalhar com a reconstrução de uma imagem sintética (o fantasma de Shepp-Logan) de tamanho 100×100, que está no moodle, no arquivo `phantom.mat`, e também pode ser encontrado no Matlab e no Octave, com o comando `phantom(100, 100)`), no Python (`from skimage.data import shepp_logan_phantom`, `pht = shepp_logan_phantom()` — o comando do Python só permite obter a imagem com dimensões 400×400) e em Julia (`using Images`, `phantom = shepp_logan(100,100)`). Você também pode ler o arquivo de dados, usando o comando `load phatom` no Matlab, ou em Julia, os comandos `using MAT; phtm = matread("phantom.mat"); phantom = phtm["I"]`.

Esta imagem representa os coeficientes de atenuação de um corte transversal de um fantasma que modela a cabeça humana. Cada pixel da imagem é um coeficiente de atenuação cujo valor varia entre 0 e 57.27, o coeficiente de atenuação do osso humano em m^{-1}. Vamos supor que a imagem tem tamanho $20cm \times 20cm$, de forma que a distância entre dois pixels adjacentes seja de $2mm$. Se você preferir usar as funções prontas para criar o fantasma, é necessário normalizar os valores dos pixels para que o máximo seja igual a $52,27$, para obter os coeficientes de atenuação.

(a) Vamos simular uma tomografia de raios paralelos com o fantasma usando uma máquina que meça 720 projeções de raios-X em ângulos linearmente espaçados, isto é, que meça uma projeção a cada $0.5^{\underline{o}}$. Considerando que as dimensões da imagem sejam de 20cm $\times$ 20cm, use a função `radon` do Matlab para calcular a atenuação total para cada projeção e depois plote o sinograma. Em Julia, é necessário usar a função `radom` do Python, com os comandos `using PyCall; simg = pyimport("skimage.transform"); sinograma = simg.radon(imagem, `θ`)`.

Não se esqueça de colocar os eixos e a escala de cor. No Matlab, `imagesc(theta,u,R)` mostra a matriz R com eixos theta e u. Os comandos `xlabel` e `ylabel` adicionam legenda aos eixos. O comando `colorbar` adiciona a escala de cores.

Em Julia, o comando `using PyPlot; imshow(imagem/maximum(imagem))` desenha a figura (a divisão por `maximum(imagem)` é necessária caso os valores dos pixels não estejam no intervalo [0, 1]). A escala de cores também é obtida em Julia com o comando `colorbar()`.

`Importante:` A função Radon considera que o espaçamento entre dois pixels é unitário. Como a integral da Transformada Radon foi trocada por uma soma, precisamos multiplicar o resultado pelo '$\mathrm{d}\ell$' equivalente, no nosso caso igual a 2 mm.

(b) Na prática, o número de fótons em cada projeção é limitado, gerando um ruído de medida do tipo *Poisson*. Se N é o número de fótons recebidos por um detector e N_0 é o número de fótons emitido pela fonte na direção do detector, então o valor esperado do número de fótons recebidos segue a lei de Beer-Lambert:

$$\mathbb{E}\{N\} = N_0 e^{-\int \mu(x,y)\mathrm{d}\ell} = N_0 e^{-R(u,\theta)},$$

onde μ é a imagem e R sua transformada Radon.

Suponha que sejam emitidos $N_0 = 100$ fótons na direção de cada detector em cada projeção. Simule o número de fótons recebidos por cada detector em cada projeção e então plote o sinograma medido. Compare com o sinograma sem ruído.

Dica: Basta gerar um ruído de Poisson N com média $N_0 e^{-R(u,\theta)}$ e depois calcular o sinograma ruidoso como $R_{noisy}(u,\theta) = -\ln(N/N_0)$. Em Matlab, use a função `poissrnd` para gerar amostras de uma variável de Poisson, em Julia, os comandos `using Distributions`; `rand(Poisson(λ))` geram uma amostra de uma variável de Poisson com média λ.

(c) Reconstrua as imagens com ruído e sem ruído usando a transformada Radon inversa sem a aplicação do filtro. Para isso, em Matlab use a função `iradon` com o par de argumentos 'Filter', 'None'. Depois, reconstrua usando o filtro de Ram-Lak. Em Julia e Python, o comando `iradon` assume o filtro de Ram-Lak como default. Para não usar filtro algum, use `iradon(sinograma, θ, filter_name = nothing)` em Julia (`filter_name = None` em Python).

Compare as quatro imagens que você obteve.

2. O Filtered Back Projection é uma das dezenas de técnicas de reconstrução de imagens tomográficas. Uma outra classe de algoritmos de reconstrução consiste em modelar o problema de reconstrução tomográfica como

$$\mathbf{y} = \mathbf{A}\mathbf{x}$$

onde $\mathbf{y}$ é a matriz de projeções vetorizada (uma coluna em cima da outra), $\mathbf{x}$ é a o mapa de atenuações vetorizado e $\mathbf{A}$ é uma matriz característica do aparelho de tomografia que pode ser modelada medindo o número de fótons recebidos por cada sensor quando não há nenhum objeto atenuante. Uma das vantagens de algoritmos deste tipo é que não é necessário considerar que os raios são paralelos como no caso da FBP, pois a geometria dos raios está descrita na matriz A.

O arquivo 'walnut.mat' contém as projeções da tomografia de uma noz, retiradas de https://www.fips.fi/dataset.php. Neste caso, os feixes de raios-X não são exatamente paralelos, mas a matriz $\mathbf{A}$ é fornecida.

Você aprendeu em Cálculo Numérico a resolver o sistema sobredeterminado $\mathbf{y} = \mathbf{A}\mathbf{x}$. Uma das maneiras é por mínimos quadrados, isto é, encontrar $\mathbf{x}$ que minimiza a função-custo

$$J(\mathbf{x}) = \|\mathbf{y} - \mathbf{A}\mathbf{x}\|_2^2$$

No entanto, sabemos que a imagem vetorizada $\mathbf{x}$ não pode ter coeficientes muito grandes. Por isso, é comum adicionar um regularizador à função-custo que limita a norma de $\mathbf{x}$. Esta técnica é chamada de Regularização de Tikhonov, e consiste em encontrar $\mathbf{x}$ que minimiza a função-custo

$$J(\mathbf{x}) = \|\mathbf{y} - \mathbf{A}\mathbf{x}\|_2^2 + \lambda \|\mathbf{x}\|_2^2 ,$$

em que $\lambda \geq 0$ é uma constante que penaliza a norma de $\mathbf{x}$. É possível mostrar que o valor de $\mathbf{x}$ que minimiza $J(\mathbf{x})$ é dado por

$$\mathbf{x} = (\mathbf{A}^T\mathbf{A} + \lambda\mathbf{I})^{-1}\mathbf{A}^T\mathbf{y}$$

Aplique a transformada Radon inversa com o filtro de Ram-Lak às projeções da noz e compare com a solução obtida pela regularização de Tikhonov com os valores $\lambda = 0$, $\lambda = 0.1$, $\lambda = 1$, $\lambda = 10$, $\lambda = 10^2$, $\lambda = 10^3$, $\lambda = 10^4$, $\lambda = 10^5$ e $\lambda = 10^6$.

Na sua opinião, qual é o melhor valor de λ neste caso?

`Dica`: O arquivo walnut.mat contém duas variáveis: $\mathbf{A}$ e as projeções $\mathbf{y}$. Cada projeção foi medida em 120 ângulos linearmente espaçados entre 0 e 357º, e com 82 sensores para cada projeção. No Matlab pode usar `y(:)` para vetorizar a matriz y (em Julia, `y[:]`), e depois usar a função `vec2mat` para transformar a imagem reconstruída vetorizada em uma matriz (em Julia, use a função `reshape(x, Nx, Nx)`).