Capítulo 4

Implementação de convoluções

Um dos detalhes mais irritantes dos espaços de escala (e em geral de Processamento de Sinais) é que a teoria está feita para sinais contínuos ou discretos infinitos enquanto os sinais do processamento computacional são em geral finitos e discretos. Neste apêndice, gostaríamos de citar as diferentes maneiras de lidar com este problema e esboçar os algoritmos que implementam as convoluções discretas correspondentes.

No que se segue, discutiremos o caso unidimensional (mais uma vez, o caso multidimensional é análogo). Assim, dado um sinal $f : I_N = \{0, 1, ..., N-1\} \to \mathbb{R}$ e um núcleo de convolução $T : \mathbb{Z} \to \mathbb{R}$, como dar sentido à expressão "$f * T$"? Como calculá-la, pelo menos nas posições originais de f, isto é, em I_N? Como implementá-las?

4.1 Extensões de f

Podemos interpretar $f * T$ como $g * T$ onde $g : \mathbb{Z} \to \mathbb{R}$ é uma boa extensão de f. A maneira mais comum de fazer isto é tomar $g \equiv f$ em I_N e então estender f de uma das seguintes formas:

- **Periodicamente**: simplesmente tome g periódica de período N

$$g[n] = f_{n \bmod N}$$

- **Por espelhamento**: para evitar descontinuidades nas bordas do sinal, concatene uma reflexão de f a uma de suas bordas e só então use a extensão

periódica

$$g[n] = \begin{cases} f_n \text{ para } 0 \leq n \leq N-1 \\ f_{(2N-1)-n} \text{ para } N \leq n \leq 2N-1 \\ g[n \bmod 2N], \text{ caso contrário} \end{cases}$$

(bordas repetidas – período $2N$)

$$g[n] = \begin{cases} f_n \text{ para } 0 \leq n \leq N-1 \\ f_{(2N-2)-n} \text{ para } N \leq n \leq 2N-3 \\ g[n \bmod (2N-2)], \text{ caso contrário} \end{cases}$$

(bordas não repetidas – período $2N-2$)

- **Por constante**: simplesmente preencha g com um valor constante independente de f (frequentemente tomado como zero):

$$g[n] = \begin{cases} f_n \text{ para } 0 \leq n \leq N-1 \\ C\,(=0), \text{ caso contrário} \end{cases}$$

- **Por média**: preencha g com o valor da média de f

$$g[n] = \begin{cases} f_n \text{ para } 0 \leq n \leq N-1 \\ \mu_f = \frac{1}{N}\sum_{j=0}^{N-1} f_j, \text{ caso contrário} \end{cases}$$

Outras possibilidades incluem fazer com que haja um decaimento de f além de seus bordos em direção a algum valor constante (0 ou a média de f, em geral).

4.2 Algoritmos para convoluções

É frequente que T seja um núcleo de convolução de suporte pequeno, digamos, com m elementos (como por exemplo a maioria dos núcleos utilizados para derivadas discretas). Neste caso, a melhor maneira é simplesmente usar a definição explícita da convolução[1]

$$(g * T)[j] = \sum_{k=0}^{m-1} T_k g_{j-k}$$

que envolverá $o(m)$ operações para cada j; em geral, só interessam os índices $0 \leq j \leq N-1$, e a convolução como um todo é calculada em $o(mN)$ operações. Note que apenas $N+m-1$ elementos de g são necessários para este cálculo.

Caso T tenha suporte grande (ou, como acontece frequentemente, infinito), um dos seguintes métodos deve ser utilizado.

[1] A expressão assume que T tem 0 como primeiro índice; caso contrário, basta transladar a resposta.

4.2.1 Extensão periódica

Se, por "sorte", o núcleo de convolução T também for periódico de mesmo período que f (por exemplo, quando T for outro sinal ou imagem), podemos interpretar $f * T$ simplesmente como a convolução circular $f *_c T$ onde T é restrito a um de seus períodos. A implementação pode ser feita diretamente do somatório que define a convolução circular ou mais eficientemente com o auxílio da Transformada de Fourier Discreta (comentário 3.9)

$$(f *_c T)_j = \frac{1}{\sqrt{N}} \sum_{k=0}^{n-1} f_k T_{k \bmod n} \quad o\left(N^2\right) \text{ operações}$$

$$f *_c T = \sqrt{N}\, IFFT\left(FFT\left(f\right) \cdot FFT\left(T\right)\right) \quad o\left(N \log N\right) \text{ operações}$$

Aliás, mesmo que T seja infinito e não periódico, $g * T$ será periódica e só precisa ser calculada em I_n. De fato, de acordo com a proposição 3.11 um período de $g * T$ é dado por

$$\text{“}f * T\text{”} = IFFT\left(FFT\left(f\right) \cdot \hat{T}\left(\frac{w}{N}\right)\right)$$

onde a multiplicação é tomada componente a componente, indexadas por w. Por exemplo, para o núcleo de Poisson simétrico $P_t\left[n\right]$ (que é uma versão discreta da Gaussiana cuja DTFT é $\hat{P}_t\left(w\right) = e^{-4t\sin^2(\pi w)}$):

$$\text{“}f * P_t\text{”} = IFFT\left(e^{-4t\sin^2(\pi w/N)} \cdot FFT\left(f\right)\right) \quad o\left(N \log N\right) \text{ operações}$$

$$\left(f * P_t\right)_n = \frac{1}{N} \sum_{w,k=0}^{N-1} f_k \exp\left(\frac{2\pi i w\left(n-k\right)}{N} - 4t\sin^2\left(\frac{\pi w}{N}\right)\right)$$

onde a última expressão permite os cálculos dos coeficientes em f_k a priori e é boa para processamento paralelo.

4.2.2 Extensão por espelhamento

Simplesmente use um dos períodos de g (de tamanho $2N$ ou $2N-2$) no processo descrito na última subseção. Usando o algoritmo FFT para as Transformadas de Fourier Discretas, são realizadas $o\left(N \log N\right)$ operações.

4.2.3 Extensão por zeros

A convolução pode ser calculada diretamente através da definição da convolução de maneira semelhante à que usamos quando o núcleo era de suporte finito e pequeno

$$\left(g * T\right)\left[j\right] = \sum_{k=0}^{N-1} T_{j-k} f_k$$

Note que $g * T$ não tem suporte finito. Se desejarmos apenas os seus valores em I_N, usaremos $o\left(N^2\right)$ operações.

O cálculo de $g * T$ em I_N necessita dos valores de T_j apenas para $2N-1$ valores de j, a saber, $-N+1 \leq j \leq N-1$. Assim, pode-se encarar $g * T$ como uma convolução entre dois núcleos de suporte finito e usar o comentário 3.10: seja $\tilde{g}$ o sinal finito de tamanho $3N-2$ obtido estendendo os N valores de f com zeros; seja $\tilde{T}$ o sinal finito obtido estendendo os $2N-1$ valores de T acima com zeros:

$$\tilde{g} = \left(f_0\ f_1\ f_2\ ...\ f_{N-1}\ \underbrace{0\ 0\ 0\ ...\ 0}_{2N-2\ \text{zeros}} \right);\ \ \tilde{g}_0 = f_0$$

$$\tilde{T} = \left(T_{-N+1}\ T_{-N+2}\ ...\ T_{N-1}\ \underbrace{0\ 0\ 0\ ...\ 0}_{N-1\ \text{zeros}} \right);\ \ \tilde{T}_0 = T_{-N+1}$$

Então, $g * T$ e $\tilde{g} *_c \tilde{T}$ coincidem em N de seus valores, isto é

$$(g * T)[n] = \left(\tilde{g} *_c \tilde{T}\right)_{N-1+n} \text{ para } 0 \leq n \leq N-1$$

$$g * T \approx \hat{g} *_c \tilde{T} = \sqrt{3N-2}\ IFFT\left(FFT\left(\tilde{g}\right) \cdot FFT\left(\tilde{T}\right)\right)$$

que também pode ser feiro em $o\left(N \log N\right)$ operações.

4.2.4 Extensão por média ou outro valor constante

Se g é obtido a partir de f usando uma extensão constante (digamos C, onde possivelmente $C = \mu_f$), então

$$g = h + C\mathbb{I}$$

onde h é obtida a partir de $f - C$ estendendo-o por zeros, e $\mathbb{I}$ é a seqüência constante igual a 1:

$$h = (...0\ 0\ 0\ 0\ f_0 - C\ f_1 - C\ f_2 - C\ ...\ f_{N-1} - C\ 0\ 0\ 0...)$$
$$\mathbb{I} = (...1\ 1\ 1\ 1...)$$

Assim,

$$g * T = h * T + C\mathbb{I} * T = h * T + \left(C \sum_{k=-\infty}^{\infty} T_k\right)\mathbb{I}$$

onde $h * T$ pode ser calculado em $o\left(N \log N\right)$ operações aplicando o método da subseção anterior a $f - C$. Em particular, para núcleos normalizados (um caso comum), tem-se

$$\text{“}f * T\text{”} = g * T = (f - C) * T + C$$

4.2.5 Núcleos Recursivos

Existem núcleos de suporte infinito cuja convolução pode ser implementada de maneira recursiva. Por exemplo, o núcleo geométrico (normalizado)

$$T = \begin{pmatrix} 1 & \beta & \beta^2 & \dots & \beta^n & \dots \end{pmatrix} \cdot (1-\beta)$$

tem suporte infinito. Para obter

$$f_{out} = f_{in} * T$$

note que

$$\begin{aligned} \varphi_{out} = \varphi_{T*f_{in}} = \varphi_T \, \varphi_{in} = \frac{1-\beta}{1-\beta z} \, \varphi_{in} \Rightarrow \\ \Rightarrow (1-\beta z) \sum f_{out}[n] \, z^n = (1-\beta) \sum f_{in}[n] \, z^n \Rightarrow \\ \Rightarrow f_{out}[n] - \beta f_{out}[n-1] = (1-\beta) \, f_{in}[n] \Rightarrow \\ \Rightarrow f_{out}[n] = \beta f_{out}[n-1] + (1-\beta) \, f_{in}[n] \end{aligned}$$

nos dá uma **relação recursiva simples** para o cálculo de f_{out}!

Que outros núcleos possuem propriedades semelhantes? Em geral, podemos calcular $f_{out} = f_{in} * T$ a partir de uma relação recursiva da esquerda para a direita do tipo

$$f_{out}[n] = \sum_{j=1}^{B} b[j] \, f_{out}[n-j] + \sum_{j=-A}^{A} a[j] \, f_{in}[n-j] \tag{4.1}$$

se e somente se a função geradora do núcleo T em questão pode ser escrita como um quociente de polinômios[2]

$$\varphi_T = \frac{\sum_{-A}^{A} a_j z^j}{1 - \sum_{1}^{B} b_j z^j}$$

Definição 4.1 *Um núcleo T é* RECURSIVO *quando $f_{out} = T * f_{in}$ corresponde à relação recursiva 4.1.*

Que núcleos são recursivos? Se quebrarmos o quociente de polinômios de φ_T em frações parciais, esperamos encontrar

$$\varphi_T = \sum_j \frac{a_j z^{k(j)}}{1 - \beta_j z}$$

[2]Para ser exato, o numerador é uma série de potências finita. Note que uma relação recursiva da direita para a esquerda também pode ser escrita desta forma! Por exemplo, a relação recursiva $g_n = g_{n+1} + 3g_{n+2} + 6f_n + 3f_{n+1}$ pode ser reescrita como $g_{n+2} = -\frac{1}{3}g_{n+1} + \frac{1}{3}g_n - 2f_n - f_{n+1}$. O leitor deve se convencer de que a direção da recursão é irrelevante na análise que se segue.

ou seja, T é uma **soma** de núcleos semelhantes ao já apresentado núcleo geométrico $T_\beta = \left(1 \;\; \beta \;\; \beta^2 \;\; ... \;\; \beta^n \;\; ...\right)$, cada um possivelmente amplificado por $a/\left(1-\beta\right)$ e transladado k posições.

Ignore a amplificação e a translação por enquanto; note que podemos ter β complexo, digamos $\beta = re^{i\theta}$; neste caso, reescrevemos o núcleo acima como

$$T_\beta = \left(1 \;\; r\cos\theta + ir\sin\theta \;\; r^2\cos 2\theta + ir^2\sin 2\theta ...\right)$$

As únicas combinações lineares de tais núcleos que produzem coeficientes reais correspondem às combinações lineares **reais** dos seguintes núcleos

$$T_c = \frac{T_\beta + T_{\bar{\beta}}}{2} = \left(1 \;\; r\cos\theta \;\; r^2\cos 2\theta \;\; r^3\cos 3\theta \;\; ...\right) \Rightarrow$$
$$\Rightarrow T_c\left(n\right) = r^n\cos n\theta,\; n \geq 0$$

$$T_s = \frac{T_\beta - T_{\bar{\beta}}}{2i} = \left(0 \;\; r\sin\theta \;\; r^2\sin 2\theta \;\; r^3\sin 3\theta \;\; ...\right) \Rightarrow$$
$$\Rightarrow T_s\left(n\right) = r^n\sin n\theta,\; n \geq 0$$

As Transformadas Z correspondentes são

$$\varphi_c = \frac{\varphi_\beta + \varphi_{\bar{\beta}}}{2} = \frac{1}{2}\left(\frac{1}{1-\beta z} + \frac{1}{1-\bar{\beta}z}\right) = \frac{1-\left(r\cos\theta\right)z}{1-\left(2r\cos\theta\right)z + r^2z^2}$$
$$\varphi_s = \frac{\varphi_\beta - \varphi_{\bar{\beta}}}{2i} = \frac{1}{2i}\left(\frac{1}{1-\beta z} - \frac{1}{1-\bar{\beta}z}\right) = \frac{\left(r\sin\theta\right)z}{1-\left(2r\cos\theta\right)z + r^2z^2}$$

Resumindo, temos 3 núcleos recursivos básicos reais (aqui, normalizados) cujas funções geradoras, médias, variâncias e correspondentes relações recursivas para $f_{out} = T * f_{in}$ estão listadas abaixo:

- Geométrico

$$T = \left(...0 \;\; 1 \;\; \beta \;\; \beta^2 \;\; \beta^3 \;\; ...\right).\left(1-\beta\right)$$
$$\varphi = \frac{1-\beta}{1-\beta z};\; \mu = \frac{\beta}{1-\beta};\; \sigma^2 = \frac{\beta}{1-\beta}$$
$$f_{out}\left[n\right] = \beta f_{out}\left[n-1\right] + \left(1-\beta\right)f_{in}\left[n\right]$$

- Cosseno recursivo

$$T_c = \left(1 \;\; r\cos\theta \;\; r^2\cos 2\theta \;\; r^3\cos 3\theta \;\; ...\right).\frac{1-2r\cos\theta+r^2}{1-r\cos\theta}$$
$$\varphi_c = \frac{1-\left(r\cos\theta\right)z}{1-\left(2r\cos\theta\right)z+r^2z^2}\,\frac{1-2r\cos\theta+r^2}{1-r\cos\theta}$$
$$\mu_c = r\frac{\left(1+r^2\right)\cos\theta - 2r}{\left(1-2r\cos\theta+r^2\right)^2}$$

$$f_{out}[n] = \left((2r\cos\theta)\, f_{out}[n-1] - r^2 f_{out}[n-2]\right) + \\ + \frac{1-2r\cos\theta+r^2}{1-r\cos\theta}\left(f_{in}[n] - (r\cos\theta)\, f_{in}[n-1]\right)$$

- Seno recursivo

$$T_s = \left(r\sin\theta \;\; r^2\sin 2\theta \;\; r^3\sin 3\theta \;\; ...\right).\frac{1-2r\cos\theta+r^2}{r\sin\theta}$$
$$\varphi_s = \frac{1-2r\cos\theta+r^2}{1-(2r\cos\theta)\,z+r^2z^2}$$
$$\mu_s = \frac{1-r^2}{1-2r\,(\cos\theta)+r^2}$$

$$f_{out}[n] = \left((2r\cos\theta)\, f_{out}[n-1] - r^2 f_{out}[n-2]\right) + \\ + \left(1-2r\cos\theta+r^2\right) f_{in}[n]$$

Qualquer núcleo recursivo será uma soma de núcleos dos tipos descritos acima (possivelmente transladados e amplificados individualmente). Note que o método exige que se estabeleçam valores de f_{in} para iniciar a recursão, o que só pode ser feito diretamente se f_{in} é uma extensão por zeros ou pela constante C. Se T é normalizado, basta tomar os valores de f_{out} para iniciar a recursão como C; caso contrário

$$f_{out}[n] = C\sum_{j=-\infty}^{\infty} T[j] = C\frac{\sum_j a_j}{1+\sum_j b_j} \text{ para } n<0$$

onde $b[j]$ e $a[j]$ são como em 4.1. Caso f_{in} seja um outro tipo de extensão, algoritmos mais sofisticados são necessários.

Na prática, o método recursivo providenciará uma maneira eficientíssima de calcular a convolução $f_{in} * T$, com número de operações por índice aproximadamante igual ao número de coeficientes a_j e b_j não-nulos.

Mesmo que o núcleo de convolução a ser usado não seja recursivo, podemos aproximá-lo por um núcleo recursivo e então usar este método. Por exemplo, de acordo com [Deriche], uma amostragem da Gaussiana G_t pode ser bem aproximada utilizando uma das seguintes expressões

$$e^{-\frac{1}{4t}x^2} \approx e^{-0.891\frac{|x|}{\sqrt{t}}}\left(0.9629\cos\left(0.5974\frac{|x|}{\sqrt{t}}\right) + 1.942\sin\left(0.5974\frac{|x|}{\sqrt{t}}\right)\right)$$

$$e^{-\frac{1}{4t}x^2} \approx 1.898e^{-1.1\frac{|x|}{\sqrt{t}}} \\ - e^{-1.069\frac{|x|}{\sqrt{t}}}\left(0.8929\cos\left(1.043\frac{|x|}{\sqrt{t}}\right) - 1.021\sin\left(1.043\frac{|x|}{\sqrt{t}}\right)\right)$$

$$e^{-\frac{1}{4t}x^2} \approx e^{-1.261\frac{|x|}{\sqrt{t}}}\left(1.68\cos\left(0.4468\frac{|x|}{\sqrt{t}}\right)+3.735\sin\left(0.4468\frac{|x|}{\sqrt{t}}\right)\right) - e^{-1.218\frac{|x|}{\sqrt{t}}}\left(0.6803\cos\left(1.412\frac{|x|}{\sqrt{t}}\right)+0.2598\sin\left(1.412\frac{|x|}{\sqrt{t}}\right)\right)$$

A primeira permite que se escreva o núcleo Gaussiano G_t como a soma de um par de núcleos cosseno recursivos (possivelmente amplificado; $r = e^{-0.891/\sqrt{t}}$ e $\theta = 0.5974/\sqrt{t}$; o primeiro será para $x \geq 0$ e o outro será seu simétrico) com um par de seno recursivos (amplificado; $r = e^{-0.891\sqrt{t}}$ e $\theta = 0.5974/\sqrt{t}$), menos uma constante (para compensar pela dupla incidência no ponto $x = 0$); a segunda dá um par de cada um dos três tipos; a terceira dá dois pares de cossenos e dois pares de senos.

4.3 Bibliografia

Rachid Deriche, *Recursively implementing the gaussian and its derivatives*, Tech. Report 1893, INRIA – Institut National de Recherche en Informatique et en Automatique, 2004 route des Lucioles, BP93, 06902 Sophia-Antipolis, France, April 1993, Programme 4, Robotique, Image et Vision.