Time, Images and Graphs

Alejandro Ribeiro

We can define convolutions for time signals, convolutions for images, and convolutions on graphs. The latter is the one that is of interest to us and is defined as

$$\mathbf{y} = \sum_k \mathbf{S}^k \mathbf{x}\,, \tag{1}$$

where $\mathbf{x}$ is an input signal, $\mathbf{y}$ is an output signal, the h_k are filter coefficients, and $\mathbf{S}$ is a matrix representation of the graph. In this post, we explore connections between (1) and the usual definitions of convolutions for time signals and images.

1 Convolutions in Time and on Graphs

The convolution in time, as usually defined in signal processing courses, is equivalent to the graph convolution when particularized to a line graph. To see that this is true, let $\mathbf{x} = [x_0; \ldots; x_{N-1}]$ be an input signal and $\mathbf{y} = [y_0; \ldots; y_{N-1}]$ the output of a convolutional filter with coefficients h_k applied to input $\mathbf{x}$. As per the usual definition of convolution, the output signal $\mathbf{y}$ is the one with entries

$$y_n = \sum_k h_k x_{n-k}\,, \tag{2}$$

where we adopt the convention $x_{n-k} = 0$ if the index $n-k$ is outside the range of valid indexes $\{0, N-1\}$.

An alternative way of writing (2) is to define a time shift operator $\mathcal{S}$. This is a linear operator that when applied to signal $\mathbf{x}$ produces the signal $\mathcal{S}\mathbf{x}$ with entries

$$(\mathcal{S}\mathbf{x})_n = x_{n-1}\,. \tag{3}$$

Adopting the convention that $\mathcal{S}^k$ denotes composition of the operator $\mathcal{S}$, we can write $x_{n-k} = (\mathcal{S}^k\mathbf{x})_n$. Substituting this fact into (2) yields

$$y_n = \sum_k (\mathcal{S}^k\mathbf{x})_n\,. \tag{4}$$

The beauty of (4) is that it is an elementwise relationship. The nth component of the vector $\mathbf{y}$ is related to the nth component of the vector $\mathcal{S}^k\mathbf{x}$. We can therefore just write

$$\mathbf{y} = \sum_k \mathcal{S}^k\mathbf{x}\,. \tag{5}$$

To go from (5) to the definition of convolutions on graphs, we just need to instantiate the operator $\mathcal{S}$ as the adjacency matrix $\mathbf{S}$ of a directed line graph. That is, we just have to note that if we multiply $\mathbf{x}$ with the adjacency matrix of a line graph we produce a signal with components $[\mathbf{S}\mathbf{x}]_n = x_{n-1}$. Thus, we can equivalently write (5) as

$$\mathbf{y} = \sum_k \mathbf{S}^k\mathbf{x}\,, \tag{6}$$

which is the definition of a graph convolution.

All of this is a little pedantic. It is, for the most part, a matter of definitions. At the end of the day, the only point that matters to show the equivalency of (2) and (6) is that the product $\mathbf{S}\mathbf{x}$ is such that $[\mathbf{S}\mathbf{x}]_n = x_{n-1}$ when $\mathbf{S}$ is the adjacency matrix of a line graph.

For those of you that know representation theory, this pedantry should sound interesting. It tells you that graph signal processing is deeply related to representation theory. The expressions in (5) and (6) are different instantiations of the algebra of polynomials on the vector space of signals.

2 Convolutions on Images and Graphs

The convolution on images, as usually defined, is *not* equivalent to the graph convolution particularized to a grid graph. There is an interesting connection, however. Let $\mathbf{x}$ be an input image with pixels x_{mn} and $\mathbf{y}$ be the output of a convolution whose pixels we denote as y_{mn}. An image convolution requires filter coefficients h_{kl} with two indexes and is given by

$$y_{mn} = \sum_{k,l} h_{kl} x_{(m-k)(n-l)} \,, \tag{7}$$

where we adopt the convention $x_{(m-k)(n-l)} = 0$ if either the index $m-k$ or the index $n-l$ is outside the range of valid indexes $\{0, N-1\}$. This is straightforward generalization of (2) to two dimensions. Given that we can move in the vertical and horizontal directions, we incorporate shifting in the vertical and horizontal direction.

It is ready to see that we can repeat the steps in (3)-(5) if we define two shift operators instead of one. To make this clear let $\mathcal{Q}$ be a vertical shift operator and $\mathcal{R}$ a horizontal shift operator. These operator produce images whose pixels are shifted in the respective directions

$$(\mathcal{Q}\mathbf{x})_{mn} = x_{(m-1)n} \,, \qquad (\mathcal{R}\mathbf{x})_{mn} = x_{m(n-1)} \,. \tag{8}$$

We can now rewrite shifting using these operators and conclude that the image convolution in (7) is equivalent to

$$\mathbf{y} = \sum_{k,l} h_{kl} \mathcal{Q}^k \mathcal{R}^l \mathbf{x} \,. \tag{9}$$

This form of the image convolution is not much different from (4). The difference is that (4) is a polynomial of a single variable, the operator $\mathcal{S}$, whereas (9) is a polynomial on two variables, the operators $\mathcal{Q}$ and $\mathcal{R}$. This is as it should be, since we have two different ways of doing shifting on images.

The challenge in drawing a connection to (1) is that we cannot pigeonhole the 2-variable polynomial in (9) into the 1-variable polynomial in (1). Thus, we cannot write arbitrary image convolutions as convolutions on

graphs. This can be done if we restrict the filter coefficients h_{kl} in (7) to have some specific form. In whch case we will end up with a convolution on a grid graph. But this is besides the point here.

The interesting point we can make is that our three definitions of convolutions, namely, the graph convolution in (1), the time convolution in (5) and the image convolution in (9) are all polynomials on linear operators. This suggests that there is a more abstract formulation of convolutional filters that encompasses the three of them. This is true and it is called an Algebraic filter. For those of you versant in representation theory, convolutions in time and convolutions on graphs are generated by the algebra of single variable polynomials. Convolutions on images are generated by the algebra of two variable polynomials. The incompatibility of image convolutions with graph convolutions stems from the fact that the algebra of two variable polynomials contains two generators (that we map to $\mathcal{Q}$ and $\mathcal{R}$), whereas the algebra of single variable polynomials contains one generator (that we map to $\mathbf{S}$ or $\mathcal{S}$ depending on the application).

We may get to talk about Algebraic convolutions and Algebraic neural networks at the end of the term. They are an interesting abstraction that encompasses some other architectures. They're practical applicability, beyond times, graphs, and groups has not been explored much[1].

[1]Parada-Mayorga, Ribeiro, *"Algebraic Neural Networks: Stability to Deformations"* 2020 (arxiv.org/abs/2009.01433).