Foundations for Machine Learning

L. Y. Stefanus
TU Dresden, June-July 2019
Reference

- Mauro Caresta. The Meaning of the Convolution.
The meaning of the Convolution

The convolution of two functions $x(t)$ and $h(t)$ is defined as:

$$f(t) = (x * h)(t) = \int_{-\infty}^{\infty} x(u)h(t - u)du$$

To understand the meaning of the convolution we break it down into the following steps:

1) $x$ and $h$ are given as function of a dummy variable $u$.
2) Get the mirror of the function $h$: $h(u) \rightarrow h(-u)$.
3) Add an offset $t$ which allows $h(t - u)$ to slide along the $u$ axis in the right direction, as $t$ increases.
4) The value at a fixed $t_1$ is given by the area of the curve resulted by the product of the two functions $x(u)$ and $h(t_1 - u)$ i.e.

$$f(t_1) = \int_{-\infty}^{\infty} x(u)h(t_1 - u)du$$
The following example is the convolution of two identical window-shape functions. Note that in this case \( h(-u) = h(u) \).
area = \( A^2B \)
Exercise:
Find the convolution of the following two functions: $x(t)$ and $h(t)$. 
Properties of Convolution

- **Commutative**: \( x * h = h * x \)

\[
\int_{-\infty}^{+\infty} x(u)h(t-u)du = \int_{-\infty}^{+\infty} h(w)x(t-w)dw
\]

Proof: By the substitution \( w = t-u \).

- **Associative**: \( f * (g * h) = (f * g) * h \)
  (With the assumption that all convolution integral exist.)

- **Distributive**: \( f *(g + h) = f * g + f * h \)
CONVOLUTION: THE DISCRETE CASE
The Convolution of Two Vectors

- Given two vectors \( a = (a_0, a_1, \ldots, a_{n-1}) \) and \( b = (b_0, b_1, \ldots, b_{n-1}) \) of length \( n \), the convolution of \( a \) and \( b \) is a vector \( c = a \ast b \) with \( 2n - 1 \) components, where component \( k \) is defined as

\[
c_k = \sum_{i,j: i+j=k} a_i b_j = \sum_{i=0}^{k} a_i b_{k-i}
\]

- In other words,

\[
a \ast b = (a_0 b_0, a_0 b_1 + a_1 b_0, a_0 b_2 + a_1 b_1 + a_2 b_0, \ldots, a_{n-1} b_{n-1})
\]

- Another way to think about the convolution is to picture an \( n \times n \) table whose \( (i, j) \) entry is \( a_i b_j \), like this:
and then compute the components of the convolution vector by summing along the diagonals, as shown.
Example

- $a = (1, 2, 2)$
- $b = (2, 5, 4)$
- $c = a \ast b = (a_0 b_0,$
  
  $a_0 b_1 + a_1 b_0,$
  
  $a_0 b_2 + a_1 b_1 + a_2 b_0,$
  
  $a_1 b_2 + a_2 b_1,$
  
  $a_2 b_2)$
  
  $= (2, 9, 18, 18, 8)$
Exercise

- Compute the convolution of \( a = (2,1,2,3) \) and \( b = (4,3,2,-1) \).
The convolution can be easily generalized to vectors of different lengths, \( a = (a_0, a_1, \ldots, a_{m-1}) \) and \( b = (b_0, b_1, \ldots, b_{n-1}) \).

In this more general case, \( c = a \ast b \) is defined to be a vector with \( m + n - 1 \) components, where component \( k \) is equal

\[
c_k = \sum_{(i, j): i + j = k} a_i b_j = \sum_{i=0}^{k} a_i b_{k-i}
\]

\( i < m, j < n \)
Like in the continuous case, the discrete convolution of two vectors \( a = (a_0, a_1, \ldots, a_{m-1}) \) and \( b = (b_0, b_1, \ldots, b_{n-1}) \) can be interpreted as follows:

1. Write the vector \( b \) in reverse: \( b' = (b_{n-1}, b_{n-2}, \ldots, b_0) \).
2. Slide \( b' \) into successive positions relative to vector \( a \) for each successive value of the convolution, by summing products of the corresponding values of the two vectors.

\[
\begin{align*}
   &a_0 \ a_1 \ a_2 \ a_3 \ \cdots \ a_{m-1} \\
   &b_{n-1} \ \cdots \ b_3 \ b_2 \ b_1 \ b_0 \\
   &\quad a_0 \ a_1 \ a_2 \ a_3 \ \cdots \ a_{m-1} \\
   &b_{n-1} \ \cdots \ b_3 \ b_2 \ b_1 \ b_0 \\
   &\quad a_0 \ a_1 \ a_2 \ a_3 \ \cdots \ a_{m-1} \\
   &\quad \vdots \\
   &\quad a_0 \ a_1 \ a_2 \ a_3 \ \cdots \ a_{m-1} \\
   &b_{n-1} \ \cdots \ b_3 \ b_2 \ b_1 \ b_0 \\
\end{align*}
\]

\[
\begin{align*}
   c_0 &= a_0 b_0 \\
   c_1 &= (a_0 b_1 + a_1 b_0) \\
   c_2 &= (a_0 b_2 + a_1 b_1 + a_2 b_0) \\
   &\quad \vdots \\
   c_{m+n-2} &= a_{m-1} b_{n-1}
\end{align*}
\]
Toeplitz matrix or diagonal-constant matrix is a matrix in which each descending diagonal from left to right is constant.

Example:

\[
\begin{bmatrix}
a & b & c & d & e \\
e & a & b & c & d \\
f & e & a & b & c \\
g & f & e & a & b \\
\end{bmatrix}
\]

The convolution operation can be expressed as a matrix multiplication, where one of the inputs is converted into a Toeplitz matrix.

For example, the convolution of \(a = (a_0, a_1, a_2, a_3)\) and \(b = (b_0, b_1, b_2)\) can be formulated as:
\[ c = a \ast b = \begin{bmatrix} a_0 & 0 & 0 \\ a_1 & a_0 & 0 \\ a_2 & a_1 & a_0 \\ a_3 & a_2 & a_1 \\ 0 & a_3 & a_2 \\ 0 & 0 & a_3 \end{bmatrix} \begin{bmatrix} b_0 \\ b_1 \\ b_2 \end{bmatrix} \]
In general,

c = a \ast b =

\[
\begin{bmatrix}
  a_1 & 0 & \cdots & 0 & 0 \\
  a_2 & a_1 & \cdots & \vdots & \vdots \\
  a_3 & a_2 & \cdots & 0 & 0 \\
  \vdots & a_3 & \cdots & a_1 & 0 \\
  a_{m-1} & \vdots & \cdots & a_2 & a_1 \\
  a_m & a_{m-1} & \cdots & \vdots & \vdots \\
  0 & a_m & \cdots & a_{m-2} & \vdots \\
  0 & 0 & \cdots & a_{m-1} & a_{m-2} \\
  \vdots & \vdots & \vdots & a_m & a_{m-1} \\
  0 & 0 & 0 & \cdots & a_m
\end{bmatrix}
\begin{bmatrix}
  b_1 \\
  b_2 \\
  b_3 \\
  \vdots \\
  b_n
\end{bmatrix}
\]
Exercise

- Compute the convolution of $a = (2, 2, 3, 3, 4)$ and $b = (1, 1, 2)$ using:
  1. the sliding method
  2. matrix multiplication
  3. polynomial multiplication
Applications

- Image/Signal Smoothing.
- String Matching.
- Convolutional Neural Networks.
- Etc.
# -*- coding: utf-8 -*-

Created on Sat Jul  7 20:17:15 2018

Version 1:
Compute the convolution of two sequences a and b given as lists of numbers, using the convolution formula directly

@author: L Y Stefanus
def convo1(a,b):
    na = len(a)
    nb = len(b)
    nc = na + nb - 1
    c = [0]*nc
    a = a + [0]*(nc-na)
    b = b + [0]*(nc-nb)

    for k in range(nc):
        for i in range(k+1):
            c[k] = c[k] + a[i]*b[k-i]

    return c
# -*- coding: utf-8 -*-

Created on Sat Jul  7 21:19:13 2018

# version 2:
Compute the convolution of two sequences a and b given as lists of numbers, using the multiplication of Toeplitz matrix of a and matrix of b

@author: L Y Stefanus
import numpy as np
from scipy import linalg

def convo2(a, b):
    h = np.array(a)
    padding = np.zeros(len(b) - 1, h.dtype)
    first_col = np.r_[h, padding]
    first_row = np.r_[h[0], padding]
    T = linalg.toeplitz(first_col, first_row)
    H = np.mat(T)
    g = np.mat(b).T
    hasil = H * g
    return hasil.T.tolist()[0]
Implementation in Python 3

```python
>>> a = [1,2,2]
>>> b = [2,5,4]
>>> c = convo1(a,b)
>>> print(c)
[2, 9, 18, 18, 8]

>>> a = [2,2,3,3,4]
>>> b = [1,1,2]
>>> c = convo2(a,b)
>>> print(c)
[2, 4, 9, 10, 13, 10, 8]
```