Naive tensor function implementations

Romain Gautron
r.gautron@cgiar.org

March 5, 2019

We will work with `numpy.arrays` object. Please look at your "Python For Data Science Cheat Sheet NumPy Basics" document. Put the line "`numpy.random.seed(123)`" after your package importations in order to have reproducible results.

1 Dot product

Write a function performing dot product between two vectors (1D tensors)

Test it with two random vectors X and Y of size 8 with number in $[-1, 1[$ thanks to `numpy.random.uniform()`.

hint you will need to use the built-in function `range` ; python indexing is from 0 and to $n-1$; use `numpy.arrays.shape` attribute or built-in len() function if it is 1D tensor.

tip use `assert` built-in function to check the value of a boolean in to order insure that calculus is possible each time it's necessary.

tip try to use `pdb` debugger if needed

2 Matricial product

Write a function performing matrix multiplication using your previous function.

Test it with two random 2D arrays U and W of shapes (16,4) and (4,8) using the same function and interval than for dot product.

Check that you've got the same result using `numpy.dot` function by rounding both results to the 4th decimal using `numpy.round` function and checking element-wise equality.

hint initialize your result array with zeros.

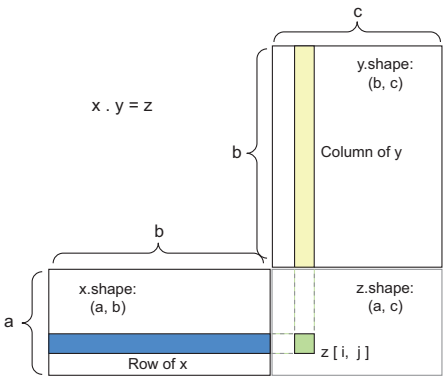

Figure 2.5 Matrix dot-product box diagram

3 Tensor addition

Write a function computing the **row-wise** addition of a first 2D tensor and a second 2D or 1D tensor. In the latter case, 1D tensor will be added multiple times.

Test it by adding to tensor W a new random vector **b** of correct length.

4 Kernel function

Thanks to your previous functions, create a function computing the kernel:

$$kernel(U, W, b) = U \cdot W + b$$

where U and W are 2D tensors and b is a vector.

tip secure the execution with correct assertions.

5 Activation function

Write a function activating the kernel with Relu function for each elements of the kernel.

$$relu(x) = \begin{cases} 0 \text{ if } x < 0, \\ x \text{ else} \end{cases}$$

6 Final tests

With same method than in section 2 check that you obtain the same result than with:

```
Knp = np.add(np.dot(U,W),b)
relu = lambda x:max(0,x)
Anp = np.vectorize(relu)(Knp) #activated kernel
```