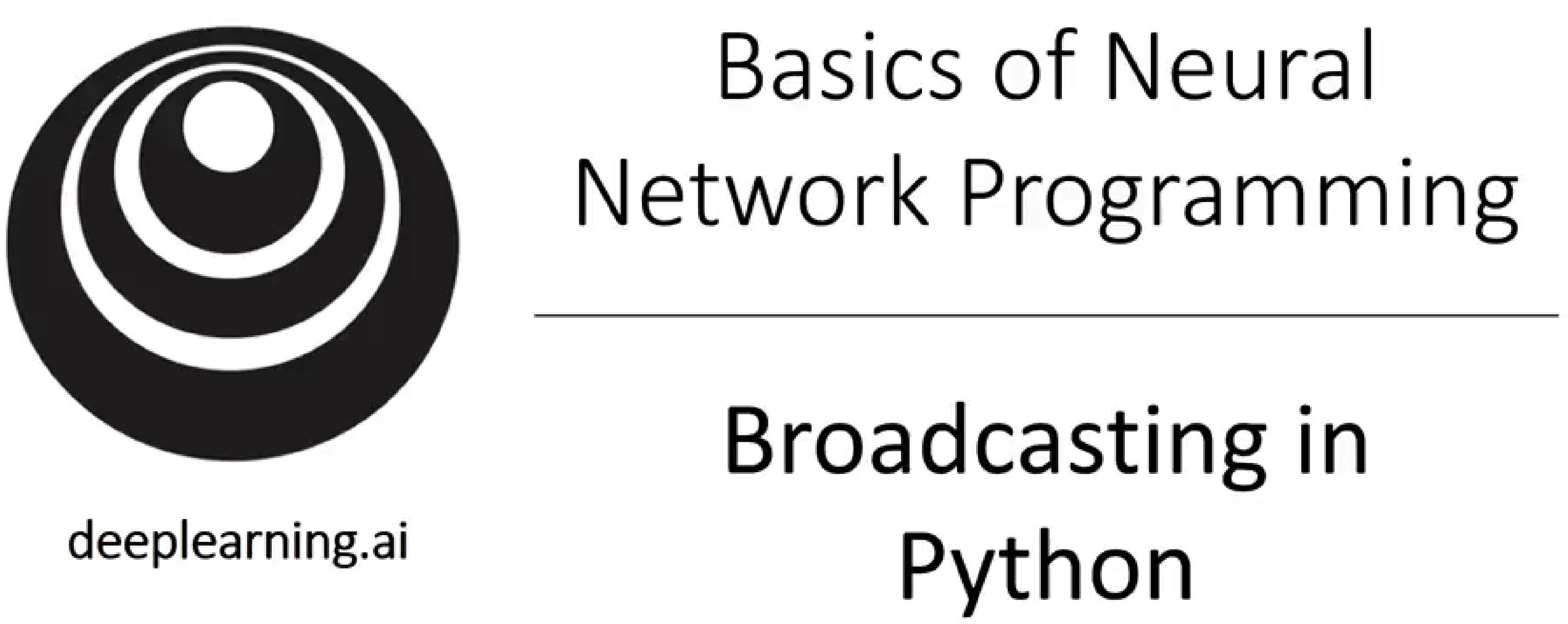

Basics of Neural Network Programming

Broadcasting in Python

deeplearning.ai

Broadcasting in Python

coursera

Have a question? Discuss this lecture in the week forums.

Interactive Transcript

English

Help us translate!

0:00
In the previous video, I mentioned that broadcasting is another technique that you can use to make your Python code run faster. In this video, let's delve into how broadcasting in Python actually works. Let's suppose today broadcasting with an example. In this matrix, I've shown the number of calories from carbohydrates, proteins, and fats in 100 grams of four different foods. So for example, a 100 grams of apples turns out, has 56 calories from carbs, and much less from proteins and fats. Whereas, in contrast, a 100 grams of beef has 104 calories from protein and 135 calories from fat. Now, let's say your goal is to calculate the percentage of calories from carbs, proteins and fats for each of the four foods. So, for example, if you look at this column and add up the numbers in that column you get that 100 grams of apple has 56 plus 1.2 plus 1.8 so that's 59 calories. And so as a percentage the percentage of calories from carbohydrates in an apple would be 56 over 59, that's about 94.9%. So most of the calories in an apple come from carbs, whereas in contrast, most of the calories of beef come from protein and fat and so on. So the calculation you want is really to sum up each of the four columns of this matrix to get the total number of calories in 100 grams of apples, beef, eggs, and potatoes. And then to divide throughout the matrix,

1:47
so as to get the percentage of calories from carbs, proteins and fats for each of the four foods. So the question is, can you do this without an explicit for-loop? Let's take a look at how you could do that.

2:04
What I'm going to do is show you how you can set, say this matrix equal to three by four matrix A. And then with one line of Python code we're going to sum down the columns. So we're going to get four numbers corresponding to the total number of calories in these four different types of foods, 100 grams of these four different types of foods. And I'm going to use a second line of Python code to divide each of the four columns by their corresponding sum. If that verbal description wasn't very clearly, hopefully it will be clearer in a

second when we look in the Python code. So here we are in the Jupiter notebook. I've already written this first piece of code to prepopulate the matrix A with the numbers we had just now, so we'll hit shift enter and just run that, so there's the matrix A. And now here are the two lines of Python code. First, we're going to compute tau equals a, that sum. And x is equals 0 means to sum vertically. We'll say more about that in a little bit. And then print cal. So we'll sum vertically. Now 59 is the total number of calories in the apple, 239 was the total number of calories in the beef and the eggs and potato and so on. And then with a compute percentage equals A/cal.reshape 1,4. Actually we want percentages, so multiply by 100 here.

3:35
And then let's print percentage.

3:40
Let's run that. And so that command we've taken the matrix A and divided it by this one by four matrix. And this gives us the matrix of percentages. So as we worked out kind of by hand just now in the apple there was a first column 94.9% of the calories are from carbs. Let's go back to the slides. So just to repeat the two lines of code we had, this is what have written out in the Jupiter notebook. To add a bit of detail this parameter, (axis = 0), means that you want Python to sum vertically. So if this is axis 0 this means to sum vertically, where as the horizontal axis is axis 1. So be able to write axis 1 or sum horizontally instead of sum vertically. And then this command here, this is an example of Python broadcasting where you take a matrix A. So this is a three by four matrix and you divide it by a one by four matrix. And technically, after this first line of codes cal, the variable cal, is already a one by four matrix. So technically you don't need to call reshape here again, so that's actually a little bit redundant. But when I'm writing Python codes if I'm not entirely sure what matrix, whether the dimensions of a matrix I often would just call a reshape command just to make sure that it's the right column vector or the row vector or whatever you want it to be. The reshape command is a constant time. It's a order one operation that's very cheap to call. So don't be shy about using the reshape command to make sure that your matrices are the size you need it to be.

5:21
Now, let's explain in greater detail how this type of operation works, right? We had a three by four matrix and we divided it by a one by four matrix. So, how can you divide a three by four matrix by a one by four matrix? Or by one by four vector?

5:40
Let's go through a few more examples of broadcasting. If you take a 4 by 1 vector and add it to a number, what Python will do is take this number and auto-expand it into a four by one vector as well, as follows. And so the vector [1, 2, 3, 4] plus the number 100 ends up with that vector on the right. You're adding a 100 to every

element, and in fact we use this form of broadcasting where that constant was the parameter b in an earlier video. And this type of broadcasting works with both column vectors and row vectors, and in fact we use a similar form of broadcasting earlier with the constant we're adding to a vector being the parameter b in logistic regression. Here's another example. Let's say you have a two by three matrix and you add it to this one by n matrix.

6:40
So the general case would be if you have some (m,n) matrix here and you add it to a (1,n) matrix. What Python will do is copy the matrix m, times to turn this into m by n matrix, so instead of this one by three matrix it'll copy it twice in this example to turn it into this. Also, two by three matrix and we'll add these so you'll end up with the sum on the right, okay? So you taken, you added 100 to the first column, added 200 to second column, added 300 to the third column. And this is basically what we did on the previous slide, except that we use a division operation instead of an addition operation.

7:34
So one last example, whether you have a (m,n) matrix and you add this to a (m,1) vector, (m,1) matrix.

7:47
Then just copy this n times horizontally. So you end up with an (m,n) matrix. So as you can imagine you copy it horizontally three times. And you add those. So when you add them you end up with this. So we've added 100 to the first row and added 200 to the second row.

8:08
Here's the more general principle of broadcasting in Python. If you have an (m,n) matrix and you add or subtract or multiply or divide with a (1,n) matrix, then this will copy it n times into an (m,n) matrix. And then apply the addition, subtraction, and multiplication of division element wise.

8:37
If conversely, you were to take the (m,n) matrix and add, subtract, multiply, divide by an (m,1) matrix, then also this would copy it now n times. And turn that into an (m,n) matrix and then apply the operation element wise. Just one of the broadcasting, which is if you have an (m,1) matrix, so that's really a column vector like [1,2,3], and you add, subtract, multiply or divide by a row number. So maybe a (1,1) matrix. So such as that plus 100, then you end up copying this real number n times until you'll also get another (n,1) matrix. And then you perform the operation such as addition on this example element-wise. And something similar also works for row vectors.

9:38

The fully general version of broadcasting can do even a little bit more than this. If you're interested you can read the documentation for NumPy, and look at broadcasting in that documentation. That gives an even slightly more general definition of broadcasting. But the ones on the slide are the main forms of broadcasting that you end up needing to use when you implement a neural network.

10:03
Before we wrap up, just one last comment, which is for those of you that are used to programming in either MATLAB or Octave, if you've ever used the MATLAB or Octave function bsxfun in neural network programming bsxfun does something similar, not quite the same. But it is often used for similar purpose as what we use broadcasting in Python for. But this is really only for very advanced MATLAB and Octave users, if you've not heard of this, don't worry about it. You don't need to know it when you're coding up neural networks in Python. So, that was broadcasting in Python. I hope that when you do the programming homework that broadcasting will allow you to not only make a code run faster, but also help you get what you want done with fewer lines of code.

10:50
Before you dive into the programming excercise, I want to share with you just one more set of ideas, which is that there's some tips and tricks that I've found reduces the number of bugs in my Python code and that I hope will help you too. So with that, let's talk about that in the next video.

Downloads

Lecture Video mp4

Subtitles (English) WebVTT

Transcript (English) txt

Lecture Slides pptx

Would you like to help us translate the transcript and subtitles into additional languages?