A note on python:numpy vectors

(DESCRIPTION)
Text, Basics of Neural Network Programming. A note on python, numpy vectors. Website, deep learning, dot, A.I.

(SPEECH)
The ability of python to allow you to use broadcasting operations and more generally, the great flexibility of the python numpy program language is, I think, both a strength as well as a weakness of the programming language.

I think it's a strength because they create expressivity of the language.

A great flexibility of the language lets you get a lot done even with just a single line of code.

But there's also weakness because with broadcasting and this great amount of flexibility, sometimes it's possible you can introduce very subtle bugs or very strange looking bugs, if you're not familiar with all of the intricacies of how broadcasting and how features like broadcasting work.

For example, if you take a column vector and add it to a row vector, you would expect it to throw up a dimension mismatch or type error or something.

But you might actually get back a matrix as a sum of a row vector and a column vector.

So there is an internal logic to these strange effects of Python.

But if you're not familiar with Python, I've seen some students have very strange, very hard to find bugs.

So what I want to do in this video is share with you some couple tips and tricks that have been very useful for me to eliminate or simplify and eliminate all the strange looking bugs in my own code.

And I hope that with these tips and tricks, you'll also be able to much more easily write bug-free, python and numpy code.

(DESCRIPTION)
New slide, Jupiter is opened through Python, numpy vectors.

(SPEECH)
To illustrate one of the less intuitive effects of Python-Numpy, especially how you construct vectors in Python-Numpy, let me do a quick demo.

Let's set a = np.random.randn(5), so this creates five random Gaussian variables stored in array a.

And so let's print(a) and now it turns out that the shape of a when you do this is this five color structure.

And so this is called a rank 1 array in Python and it's neither a row vector nor a column vector.

And this leads it to have some slightly non-intuitive effects.

So for example, if I print a transpose, it ends up looking the same as a.

So a and a transpose end up looking the same.

And if I print the inner product between a and a transpose, you might think a times a transpose is maybe the outer product should give you matrix maybe.

But if I do that, you instead get back a number.

So what I would recommend is that when you're coding new networks, that you just not use data structures where the shape is 5, or n, rank 1 array.

Instead, if you set a to be this, (5,1), then this commits a to be (5,1) column vector.

And whereas previously, a and a transpose looked the same, it becomes now a transpose, now a transpose is a row vector.

Notice one subtle difference.

In this data structure, there are two square brackets when we print a transpose.

Whereas previously, there was one square bracket.

So that's the difference between this is really a 1 by 5 matrix versus one of these rank 1 arrays.

And if you print, say, the product between a and a transpose, then this gives you the outer product of a vector, right?

And so, the outer product of a vector gives you a matrix.

So, let's look in greater detail at what we just saw here.

The first command that we ran, just now, was this.

And this created a data structure with a.shape was this funny thing (5,) so this is called a rank 1 array.

And this is a very funny data structure.

It doesn't behave consistently as either a row vector nor a column vector, which makes some of its effects nonintuitive.

So what I'm going to recommend is that when you're doing your programing exercises, or in fact when you're implementing logistic regression or neural networks that you just do not use these rank 1 arrays.

(DESCRIPTION)
The arrays are bracketed with red ink, and written, don't use.

(SPEECH)
Instead, if every time you create an array, you commit to making it either a column vector, so this creates a (5,1) vector, or commit to making it a row vector, then the behavior of your vectors may be easier to understand.

So in this case, a.shape is going to be equal to 5,1.

And so this behaves a lot like a, but in fact, this is a column vector.

And that's why you can think of this as (5,1) matrix, where it's a column vector.

And here a.shape is going to be 1,5, and this behaves consistently as a row vector.

So when you need a vector, I would say either use this or this, but not a rank 1 array.

One more thing that I do a lot in my code is if I'm not entirely sure what's the dimension of one of my vectors, I'll often throw in an assertion statement like this, to make sure, in this case, that this is a (5,1) vector.

So this is a column vector.

These assertions are really Set to execute, and they also help to serve as documentation for your code.

So don't hesitate to throw in assertion statements like this whenever you feel like.

And then finally, if for some reason you do end up with a rank 1 array, You can reshape this, a equals a.reshape into say a (5,1) array or a (1,5) array so that it behaves more consistently as either column vector or row vector.

So I've sometimes seen students end up with very hard to track because those are the nonintuitive effects of rank 1 arrays.

By eliminating rank 1 arrays in my old code, I think my code became simpler.

And I did not actually find it restrictive in terms of things I could express in code.

I just never used a rank 1 array.

And so takeaways are to simplify your code, don't use rank 1 arrays.

Always use either n by one matrices, basically column vectors, or one by n matrices, or basically row vectors.

Feel free to toss a lot of insertion statements, so double-check the dimensions of your matrices and arrays.

And also, don't be shy about calling the reshape operation to make sure that your matrices or your vectors are the dimension that you need it to be.

So that, I hope that this set of suggestions helps you to eliminate a cause of bugs from Python code, and makes the problem exercise easier for you to complete.