1. How does unsqueeze help us to solve certain broadcasting problems?

ANS:

unsqueeze turns an n.d. tensor into an (n+1).d. one by adding an extra dimension of depth 1. However, since it is ambiguous which axis the new dimension should lie across (i.e. in which direction it should be "unsqueezed"), this needs to be specified by the [dim](https://pytorch.org/docs/stable/generated/torch.unsqueeze.html) argument.

1. How can we use indexing to do the same operation as unsqueeze?

ANS:

When we squeeze a tensor, the dimensions of size 1 are removed. The elements of the original tensor are arranged with the remaining dimensions. For example, if the input tensor is of shape: (m×1×n×1) then the output tensor after squeeze will be of shape: (m×n). The following is the syntax of the torch.squeeze() method.

1. How do we show the actual contents of the memory used for a tensor?

ANS:

All built-in objects will return correct results, but this does not have to hold true for third-party extensions as it is implementation specific.

so it could be that for tensors \_\_sizeof\_\_ is undefined or defined differently than you would expect - this function is not something you can rely on. Secondly

Only the memory consumption directly attributed to the object is accounted for, not the memory consumption of objects it refers to.

which means that if the torch.Tensor object merely holds a reference to the actual memory, this won't show in sys.getsizeof. This is indeed the case, if you check the size of the underlying [storage](https://pytorch.org/docs/master/generated/torch.Tensor.storage.html#torch.Tensor.storage) instead, you will see the expected number

import torch, sys

b = torch.randn(1, 1, 128, 256, dtype=torch.float64)

sys.getsizeof(b)

>> 72

sys.getsizeof(b.storage())

>> 262208

1. When adding a vector of size 3 to a matrix of size 3×3, are the elements of the vector added to each row or each column of the matrix? (Be sure to check your answer by running this code in a notebook.)

ANS:

The real 3 by 3 matrices form a vector space M . The symmetric matrices in M form a subspace S. If you add two symmetric matrices, or multiply by real numbers, the result is still a symmetric matrix.

1. Do broadcasting and expand\_as result in increased memory use? Why or why not?

ANS:

Array broadcasting is very common in Python with numpy and in Matlab.

Broadcasting means that in an array expression if an array has a unitary dimension along an axis that dimension is automatically expanded to match the size of the other array. For example if an array has size [3,1] and another has size [1,4] both array are expanded as they have size [3,4]. In the actual Fortran one has to use spread to achieve the same result.  
Another nice feature of numpy is the possibility to add unitary dimensions on the fly with None . Python example:

a = np.ones

b= np.ones

Moreover some functions that reduce the dimensionality like sum can have an extra arguments keepdims that doesn’t reduce the dimensionality but make that dimension unitary. This is useful as one can write (in Python) somenthing like:  
a/np.mean(a, axis=0,keepdims=True)  
So I suggest the following feature:  
Use the symbol “+” (or whatever other symbol) to add a dimension to a section of an array. The function sum and other that reduce the dimensionality should have an extra keyword argument like keepdims that doesn’t eliminate that dimension.  
As an example:

integer :: a(3), b(4), c(3,4), d(4), i

a = [(i=1,3)]

b = [(i=1,4)]

c = a(:,+) + b(+,:)

d = c/sum(c, dim=1, keepdim=.true.)

1. Implement matmul using Einstein summation.

ANS:

For Matrix Vector multiplication with Einstein summation convention, use the numpy.einsum() method in Python. The 1st parameter is the subscript. It specifies the subscripts for summation as comma separated list of subscript labels. The 2nd parameter is the operands. These are the arrays for the operation.

The einsum() method evaluates the Einstein summation convention on the operands. Using the Einstein summation convention, many common multi-dimensional, linear algebraic array operations can be represented in a simple fashion. In implicit mode einsum computes these values.

In explicit mode, einsum provides further flexibility to compute other array operations that might not be considered classical Einstein summation operations, by disabling, or forcing summation over specified subscript labels.

Steps

At first, import the required libraries −

import numpy as np

Creating two numpy One-Dimensional array using the array() method −

arr1 = np.arange(25).reshape(5,5)

arr2 = np.arange(5)

Display the arrays −

print("Array1...\n",arr1)

print("\nArray2...\n",arr2)

Check the Dimensions of both the arrays −

print("\nDimensions of Array1...\n",arr1.ndim)

print("\nDimensions of Array2...\n",arr2.ndim)

Check the Shape of both the arrays −

print("\nShape of Array1...\n",arr1.shape)

print("\nShape of Array2...\n",arr2.shape)

For Matrix Vector multiplication with Einstein summation convention, use the numpy.einsum() method in Python −

print("\nResult (Matrix Vector multiplication)...\n",np.einsum('ij,Steps

At first, import the required libraries −

import numpy as np

Creating two numpy One-Dimensional array using the array() method −

arr1 = np.arange(25).reshape(5,5)

arr2 = np.arange(5)

Display the arrays −

print("Array1...\n",arr1)

print("\nArray2...\n",arr2)

Check the Dimensions of both the arrays −

print("\nDimensions of Array1...\n",arr1.ndim)

print("\nDimensions of Array2...\n",arr2.ndim)

j', arr1, arr2))

1. What does a repeated index letter represent on the lefthand side of einsum?

ANS:

The einsum function is one of NumPy’s jewels. It can often outperform familiar array functions in terms of speed and memory efficiency, thanks to its expressive power and smart loops. On the downside, it can take a little while understand the notation and sometimes a few attempts to apply it correctly to a tricky problem.

* Repeating letters between input arrays means that values along those axes will be multiplied together. The products make up the values for the output array.

In this case, we used the letter j twice: once for A and once for B. This means that we’re multiplying each row of A with each column of B. This will only work if the axis labelled by j is the same length in both arrays (or the length is 1 in either array).

There are quite a few questions on sites like Stack Overflow which about what einsum does and how it works, so this post hopes to serve as a basic introduction to the function and what you need to know to begin using it.

1. What are the three rules of Einstein summation notation? Why?

ANS:

Einstein summation is a [notational convention](https://mathworld.wolfram.com/Notation.html) for simplifying expressions including [summations](https://mathworld.wolfram.com/Sum.html) of [vectors](https://mathworld.wolfram.com/Vector.html), [matrices](https://mathworld.wolfram.com/Matrix.html), and general [tensors](https://mathworld.wolfram.com/Tensor.html). There are essentially three rules of Einstein summation notation, namely:

1. Repeated indices are implicitly summed over.

2. Each index can appear at most twice in any term.

3. Each term must contain identical non-repeated indices.

The first item on the above list can be employed to greatly simplify and shorten equations involving [tensors](https://mathworld.wolfram.com/Tensor.html). For example, using Einstein summation,

|  |
| --- |
|  |

and

|  |  |
| --- | --- |
|  |  |
|  |  |

The second and third items on the list indicate that the expression

|  |  |
| --- | --- |
|  |  |

is valid, whereas the expressions

|  |  |
| --- | --- |
|  |  |

and

|  |  |
| --- | --- |
|  |  |

are invalid because the index  appears three times in the first term of (), while the non-repeated index  in the first term of () doesn't match the non-repeated  of the second term.

1. What are the forward pass and backward pass of a neural network?

ANS:

Forward Propagation is the way to move from the Input layer (left) to the Output layer (right) in the neural network. The process of moving from the right to left i.e backward from the Output to the Input layer is called the Backward Propagation.

1. Why do we need to store some of the activations calculated for intermediate layers in the forward pass?

ANS:

Forward propagation (or forward pass) refers to the calculation and storage of intermediate variables (including outputs) for a neural network in order from the input layer to the output layer. We now work step-by-step through the mechanics of a neural network with one hidden layer. This may seem tedious but in the eternal words of funk virtuoso James Brown, you must “pay the cost to be the boss”.

For the sake of simplicity, let’s assume that the input example is and that our hidden layer does not include a bias term. Here the intermediate variable is:

where  is the weight parameter of the hidden layer. After running the intermediate variable through the activation function we obtain our hidden activation vector of length ℎ,

The hidden layer output ℎ is also an intermediate variable. Assuming that the parameters of the output layer only possess a weight of , we can obtain an output layer variable with a vector of length ℎ.

Assuming that the loss function is and the example label is , we can then calculate the loss term for a single data example,

According to the definition of ℓ2 regularization that we will introduce later, given the hyperparameter , the regularization term is

where the Frobenius norm of the matrix is simply the ℓ2 norm applied after flattening the matrix into a vector. Finally, the model’s regularized loss on a given data example is:

We refer to as the objective function in the following discussion.

1. What is the downside of having activations with a standard deviation too far away from 1?

ANS:

The disadvantages of standard deviation are : It doesn't give you the full range of the data. It can be hard to calculate.

Since this CV value is greater than 1, it tells us that the standard deviation of the data values are quite high.

The other advantage of SD is that along with mean it can be used to detect skewness. The disadvantage of SD is that it is an inappropriate measure of dispersion for skewed data.

It is not rigidly defined as it can be calculated with respect to mean, median, and mode. Sociological studies rarely use this measure to analyze data. Negative and positive signs are ignored because we take the absolute value. This can lead to inaccuracies in the result.

Disadvantages of Standard Deviation : (1) Difficult : Standard Deviation is difficult to calculate or understand. (2) More importance to Extreme Value : In the calculation of standard deviation extreme values get greater importance.

1. How can weight initialization help avoid this problem?

ANS:

The aim of weight initialization is to prevent layer activation outputs from exploding or vanishing during the course of a forward pass through a deep neural network.

Its main objective is to prevent layer activation outputs from exploding or vanishing gradients during the forward propagation.

We can try initializing this network with different methods and observe the impact on the learning.

1. Choose input dataset. Select a training dataset. ...
2. Choose initialization method. Select an initialization method for the values of your neural network parameters . ...
3. Train the network.