Using Autograd for Differetiation in Pytorch

By Sina Mohammadi

01/28/2022

1 Differentiation with Autograd in Pytorch

A PyTorch Tensor is basically the same as a numpy array, in other word it is just a generic n-dimensional array to be used for arbitrary numeric computation. In this note I use Pytorch to calculate derivatives. This note covers following topics:
2.Implementing Partial Derivatives of Functions.
3.Derivative of Functions with Multiple Values

```
import matplotlib.pyplot as plt
import torch

#### Example #1
# Declaring our 0D tensor x
x = torch.tensor(3.0 , requires_grad= True)
print(x)

#Defining Function that we want to differentiate
y= 3*x**2

# applying .bacward on y and forming acyclic graph storing the computation history,
#and evaluate the result with .grad
print("Value of function at x=3 is:",y)
y.backward()
print("Derivative of Above equation is:",x.grad)

#### Example #2
x = torch.tensor(3.0, requires_grad = True)
y = 6 * x ** 2 + 2 * x + 4
print("Result of the equation is: ", y)
y.backward()
print("Derivative of the equation at x = 3 is: ", x.grad)
```

2 Implementing Partial Derivatives of Functions.

```
#### Example #3 Partial Derivatives
u = torch.tensor(3., requires_grad=True)
v = torch.tensor(4., requires_grad=True)

f = u**2 + v**3 + 2*u*v + u + v

print(u)
print(v)
print(f)

f.backward()
print("Partial derivative with respect to u: ", u.grad)
print("Partial derivative with respect to u: ", v.grad)
```

3 Derivative of function on a domain

```
# compute the derivative of the function with multiple values
x = torch.linspace(-10, 10, 100, requires_grad = True)
Y = 10*x ** 3
y = torch.sum(Y)
y.backward()

# ploting the function and derivative
function_line, = plt.plot(x.detach().numpy(), Y.detach().numpy(), label = 'Main Function')
function_line.set_color("red")
derivative_line, = plt.plot(x.detach().numpy(), x.grad.detach().numpy(), label = 'Derivative of function')
derivative_line.set_color("green")
plt.xlabel('x')
plt.legend()
plt.show()
```

4 Attachment: What is Tensor ?

A tensor is a container which can house data in N dimensions. Often and erroneously used interchangeably with the matrix (which is specifically a 2-dimensional tensor), tensors are generalizations of matrices to N-dimensional space.

Mathematically speaking, tensors are more than simply a data container, however. Aside from holding numeric data, tensors also include descriptions

of the valid linear transformations between tensors. Examples of such transformations, or relations, include the cross product and the dot product. From a computer science perspective, it can be helpful to think of tensors as being objects in an object-oriented sense, as opposed to simply being a data structure.

5 Sources

https://pytorch.org/
https://machinelearningmastery.com/
https://www.kdnuggets.com/