COMPSCI 589
Lecture 8: Automatic Differentiation and PyTorch

Benjamin M. Marlin

College of Information and Computer Sciences
University of Massachusetts Amherst

Slides by Benjamin M. Marlin (marlin@cs.umass.edu).
Created with support from National Science Foundation Award# IIS-1350522.

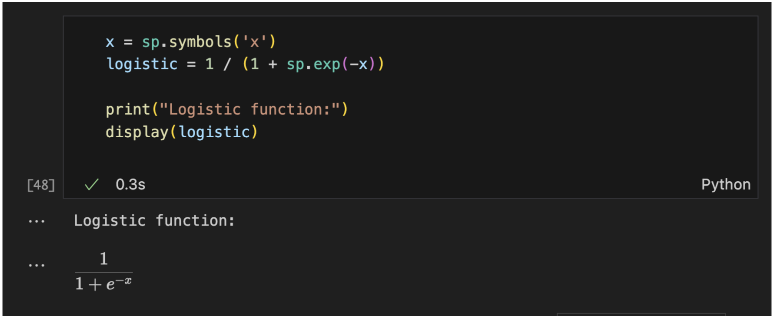
Computing Gradients

- As we have seen, a large part of the application of neural networks comes down to the computation of gradients.
- For large and complicated models, deriving gradients by hand is tedious and error-prone.
- The three options to automate the process are symbolic differentiation, numerical differentiation and automatic differentiation.

Symbolic Differentiation

- Symbolic differentiation takes as input a function $f(\theta)$ and returns a function that can compute $\nabla f(\theta)$ for any θ.
- This requires a computer algebra program or library like Wolfram Alpha, Maple, or SymPy.
- The major disadvantage of symbolic differentiation is that the function to compute the gradient can be representationally much more complex than the function whose gradient is needed.
- This means symbolic differentiation does not scale well for large and complicated function.

Example: SymPy Logistic Function



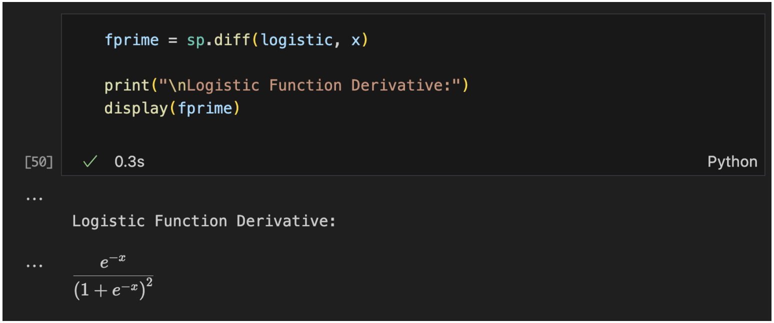
```
x = sp.symbols('x')
logistic = 1 / (1 + sp.exp(-x))

print("Logistic function:")
display(logistic)
```

```
Logistic function:
```

$$\frac{1}{1+e^{-x}}$$

Example: SymPy Logistic Function Derivative



```
fprime = sp.diff(logistic, x)

print("\nLogistic Function Derivative:")
display(fprime)
```

[50] ✓ 0.3s Python

```
Logistic Function Derivative:
```

$$\frac{e^{-x}}{\left(1 + e^{-x}\right)^{2}}$$

Numerical Differentiation

- Numerical differentiation takes as input a function $f(\theta)$ and a specific value of θ and returns an approximation to the gradient $\nabla f(\theta)$ using finite differences.

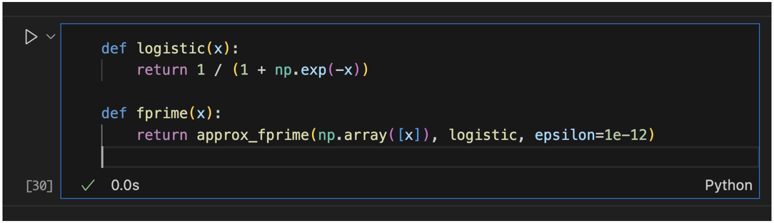
- Define a new parameter vector $\theta^{(i)}$ such that $\theta_i^{(i)} = \theta_i + \epsilon$ and $\theta_j^{(i)} = \theta_j$ for $i \neq j$. The partial derivative of $f(\theta)$ with respect to θ_i is then given by:

$$\frac{\partial}{\partial \theta_i} f(\theta) = \lim_{\epsilon \to 0} \frac{f(\theta^{(i)}) - f(\theta)}{\epsilon}$$

- To obtain a numerical approximation, we use a small value of ϵ like 10^{-8} instead of taking the limit.

- The problem with this approach is that it requires $O(K)$ objective function evaluation for a K-dimensional parameter vector θ.

Example: ScyPy ApproxFprime



```python
def logistic(x):
    return 1 / (1 + np.exp(-x))

def fprime(x):
    return approx_fprime(np.array([x]), logistic, epsilon=1e-12)
```

Automatic Differentiation

- Automatic differentiation takes as input a function $f(\theta)$ and a specific value of θ and returns the exact gradient of $\nabla f(\theta)$ at θ.
- Automatic differentiation works by computing the objective function $f(\theta)$ (the forward pass) while caching all intermediate computations.
- At the same time the forward pass is being computed, the derivatives of the functions used in the forward pass are cached.
- Finally, a backwards pass is conducted to compute the value of the gradient vector using the chain rule.

Example: Automatic Differentiation

Consider the computation of the logistic function $f(x) = 1/(1 + \exp(-x))$. The true derivative with respect to x is $\frac{\partial f(x)}{\partial x} = f(x)(1 - f(x))$. Automatic differentiation yields:

$$a = -1 * x \qquad \frac{\partial a}{\partial x} = -1$$

$$b = \exp(a) \qquad \frac{\partial b}{\partial a} = \exp(a) = b$$

$$c = 1 + b \qquad \frac{\partial c}{\partial b} = 1$$

$$f(x) = d = 1/c \qquad \frac{\partial d}{\partial c} = -\frac{1}{c^2}$$

Example: Automatic Differentiation

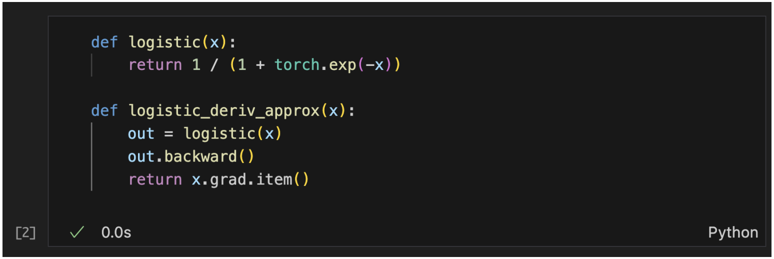
Putting the pieces together we have:

$$\begin{aligned}\frac{\partial f(x)}{\partial x} &= \frac{\partial d}{\partial c}\frac{\partial c}{\partial b}\frac{\partial b}{\partial a}\frac{\partial a}{\partial x} \\ &= \left(-\frac{1}{c^2}\right)(1)(b)(-1)\end{aligned}$$

We can re-expand the cached computations to yield:

$$\begin{aligned}\frac{\partial f(x)}{\partial x} &= \left(-\frac{1}{c^2}\right)(1)(b)(-1) \\ &= \left(-\frac{1}{(1+\exp(-x))^2}\right)(1)(\exp(-x))(-1) \\ &= f(x)(1-f(x))\end{aligned}$$

Example: PyTorch



```python
def logistic(x):
    return 1 / (1 + torch.exp(-x))

def logistic_deriv_approx(x):
    out = logistic(x)
    out.backward()
    return x.grad.item()
```

Example: PyTorch

- Modern machine learning tools like PyTorch, TensorFlow, and JAX make automatic differentiation extremely easy to use with large networks.
- To use automatic differentiation to learn model parameters, the user only needs to specify the model, the learning objective function, and the optimizer. Automatic differentiation is used to supply the gradients needed by the optimizer.