CSE 547 - Assignment 1

Philip Pham

April 17, 2018

Problem 0

List of collaborators: I have not collaborated with anyone.

List of acknowledgements: None.

Certify that you have read the instructions: Yes.

Terms and Conditions to use the dataset: I accept the terms and conditions to use the COCO dataset.

Problem 1

Read the course website, up until "Lecture Notes and Readings", so that you understand the course policies on grading, late policies, projects, requirements to pass etc. Write "I have read and understood these policies" to certify this. If you have questions, please contact the instructors.

Solution

I have read and understood these policies.

Problem 2

Consider the function from class (and the notes):

$$f\left(w_1, w_2\right) = \left[\sin\left(2\pi\frac{w_1}{w_2}\right) + 3\frac{w_1}{w_2} - \exp\left(2w_2\right)\right]\left[3\frac{w_1}{w_2} - \exp\left(2w_2\right)\right]. \tag{1}$$

Suppose our program for this function uses the following evaluation trace:

input: $z_0 = (w_1, w_2)$

1. $z_1 = w_1/w_2$
2. $z_2 = \sin\left(2\pi z_1\right)$
3. $z_3 = \exp\left(2w_2\right)$
4. $z_4 = 3z_1 - z_3$
5. $z_5 = z_2 + z_4$
6. $z_6 = z_4 z_5$

return: z_6

The forward mode of AD

The forward mode for auto-differentiation is a conceptually simpler way to compute the derivative. Let us examine the forward mode to compute the derivative of one variable, $\frac{df}{dw_1}$. In the forward mode, we sequentially compute both z_t and its derviative $\frac{dz_t}{dw_1}$ using the previous variables $z_1, \ldots, z_{t-1}$ and the previous derivatives $\frac{dz_1}{dw_1}, \ldots, \frac{dz_{t-1}}{dw_1}$.

Explicitly write out the forward mode in our example.

Solution

$$\begin{aligned}
\frac{df}{dw_1} = \frac{dz_6}{dw_1} &= \frac{\partial z_6}{\partial z_4}\frac{dz_4}{dw_1} + \frac{\partial z_6}{\partial z_5}\frac{dz_5}{dw_1} \\
&= \frac{\partial z_6}{\partial z_5}\left(\frac{\partial z_4}{\partial z_1}\frac{dz_1}{dw_1} + \frac{\partial z_4}{\partial z_3}\frac{dz_3}{dw_1}\right) + \frac{\partial z_6}{\partial z_5}\left(\frac{\partial z_5}{\partial z_2}\frac{dz_2}{dw_1} + \frac{\partial z_5}{\partial z_4}\frac{dz_4}{dw_1}\right) \\
&= \frac{\partial z_6}{\partial z_5}\left(\frac{\partial z_4}{\partial z_1}\frac{dz_1}{dw_1} + \frac{\partial z_4}{\partial z_3}\frac{dz_3}{dw_1}\right) + \frac{\partial z_6}{\partial z_5}\left(\frac{\partial z_5}{\partial z_2}\frac{dz_2}{dw_1} + \frac{\partial z_5}{\partial z_4}\left(\frac{\partial z_4}{\partial z_1}\frac{dz_1}{dw_1} + \frac{\partial z_4}{\partial z_3}\frac{dz_3}{dw_1}\right)\right)
\end{aligned}$$

Suppose we want to calculate $\frac{df}{dw_1}(u, v)$.

1. Fix $w_1 = u$ and $w_2 = v$.
2. Compute $z_1 = u/v$ and $\frac{dz_1}{w_1} = 1/v$.
3. Compute $z_2 = \sin(2\pi z_1)$ and $\frac{dz_2}{dw_1} = \frac{\partial z_2}{\partial z_1}\frac{dz_1}{w_1} = 2\pi\cos(2\pi z_1)\frac{dz_1}{w_1}$.
4. Compute $z_3 = \exp(2v)$ and $\frac{dz_3}{dw_1} = 0$.
5. Compute $z_4 = 3z_1 - z_3$ and
$$\frac{dz_4}{dw_1} = \frac{\partial z_4}{\partial z_1}\frac{dz_1}{dw_1} + \frac{\partial z_4}{\partial z_3}\frac{dz_3}{dw_1} = 3\frac{dz_1}{dw_1} - \frac{dz_3}{dw_1}.$$
6. Compute $z_5 = z_2 + z_4$ and
$$\begin{aligned}\frac{dz_5}{dw_1} &= \frac{\partial z_5}{\partial z_2}\frac{dz_2}{dw_1} + \frac{\partial z_5}{\partial z_4}\frac{dz_4}{dw_1} \\ &= \frac{dz_2}{dw_1} + \frac{dz_4}{dw_1}.\end{aligned}$$
7. Compute $z_6 = z_4 z_5$ and
$$\frac{dz_6}{dw_1} = \frac{\partial z_6}{\partial z_4}\frac{dz_4}{dw_1} + \frac{\partial z_6}{\partial z_5}\frac{dz_5}{dw_1} = z_5\frac{dz_4}{dw_1} + z_4\frac{dz_5}{dw_1}.$$

The reverse mode of AD

Now let use consider the reverse mode to compute the derivative $\frac{df}{dw}$, which is a two-dimensional vector.

Explicitly write out the reverse mode in our example.

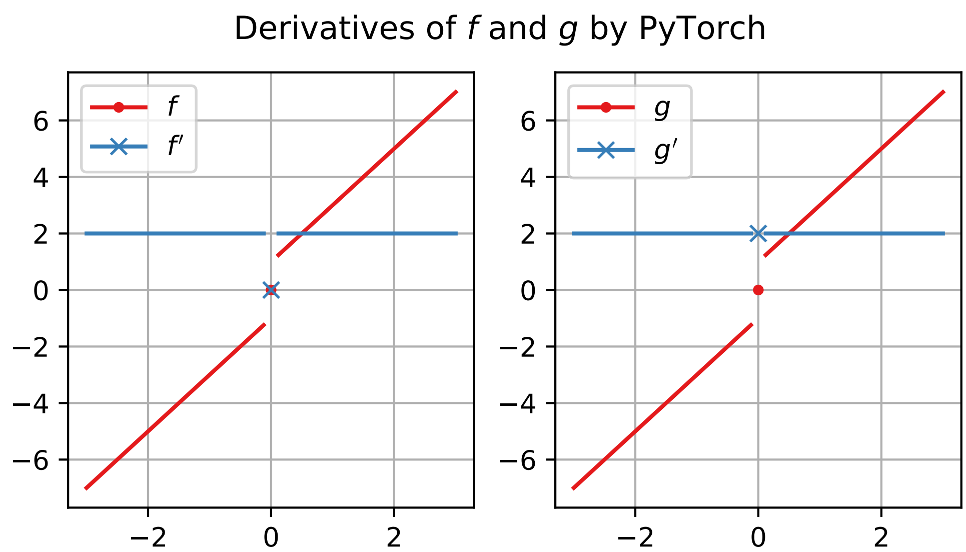

Figure 1: The functions from Listings 1 and 2 are plotted along with their PyTorch-computed derivatives.

Solution

Problem 4: PyTorch can give us some crazy answers

Now you will construct an example in which PyTorch provides derivatives that make no sense. The issue is in understanding when it is ok and when it is not ok to use dynamic computation graphs. You are going to find a way code up the same function in two different ways so that PyTorch will return different derivatives at the same point. The purpose of this exercise is to better understand how dynamic computation graphs work and to understand what you are doing when you using various AD softwares.

Two Identical Non-differentiable Functions with Different "Derivatives"

1. a

   Solution

   Consider the function

$$f(x) = \begin{cases} 2x, -1 & x < 0; \\ 0, & x = 0; \\ 2x + 1, & x > 0. \end{cases} \tag{2}$$

   It is implemented in Listings 1 and 2 and plotted in Figure 1.

2. b

   Solution

```python
def f(x: Variable) -> Variable:
    assert x.requires_grad
    return (2*x*torch.sign(x) + 1)*torch.sign(x)
```

Listing 1: Equation 2 defined with the sign function factored out.

3. c

   Solution

```python
def g(x: Variable) -> Variable:
    def g1d(x: Variable) -> Variable:
        if x.data[0] > 0:
            return 2*x + 1
        elif x.data[0] < 0:
            return 2*x - 1
        else:
            return 2*x

    if x.dim() == 0:
        return 1*x
    if x.size() == torch.Size([1]):
        return g1d(x)

    return torch.stack([g(sub_x) for sub_x in x])
```

Listing 2: Equation 2 defined element-wise by recursing into the tensor.

4. a

Extra Credit: Differentiable Functions with Different "Derivatives"

Provide a differentiable function, where you can code it up in PyTorch in two different ways and where you can get two different derivatives at the same point x_0.

Solution

We can reuse the same idea. PyTorch doesn't correctly compute the derivative when squaring the sign function, even though, the sign function squared is just the identity.

Using this, I implement the simple line $f(x) = 2x$ in two ways: (1) verbosely using the sign function (f) and (2) the canonical way (g) in Listing 3.

This results in Figure 2. PyTorch computes $f'(0) = 0$ despite the true value being 2.

```python
def f(x: Variable) -> Variable:
    assert x.requires_grad
    return 2*x*torch.sign(x)*torch.sign(x)

def g(x: Variable) -> Variable:
    assert x.requires_grad
    return 2*x
```

Listing 3: The function $f(x) = 2x$ defined in two different ways.

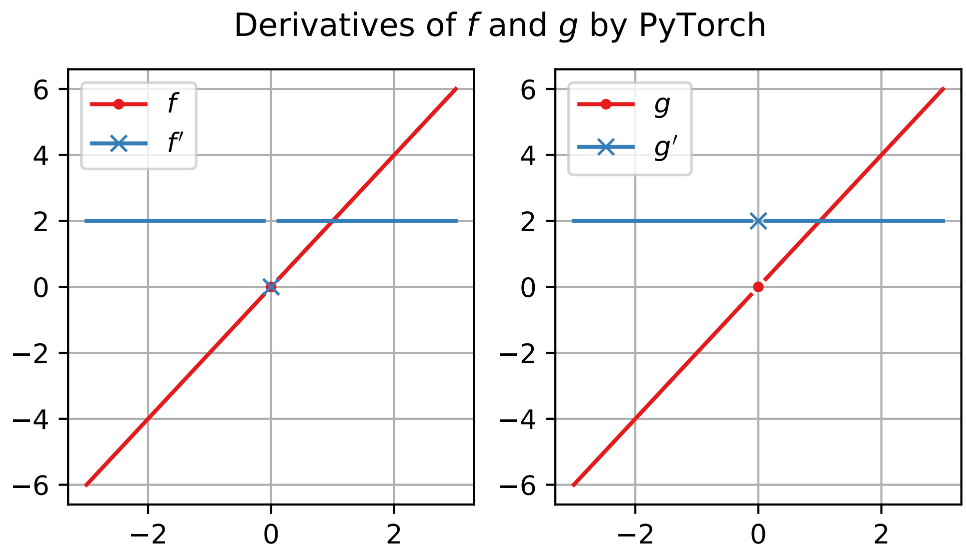

Figure 2: f and g from Listing 3 plotted along with their PyTorch-computed derivatives.

Problem 5: Elementary properties of l_2 regularized logisitic regression

The binary case

Consider minimizing

$$J(\mathbf{w}) = -l\left(\mathbf{w}, \mathcal{D}_{\text{train}}\right) + \lambda \left\|\mathbf{w}\right\|_2^2, \tag{3}$$

where

$$l\left(\mathbf{w}, \mathcal{D}\right) = \sum_j \log \mathbf{P}\left(y^j \mid \mathbf{x}^j, \mathbf{w}\right) \tag{4}$$

is the log-likelihood on the data set $\mathcal{D}$ for $y^j \in \{\pm 1\}$

State if the following are true or false. Briefly explain your reasoning.

1. With $\lambda > 0$ and the features x_k^j linearly separable, $J\left(\mathbf{w}\right)$ has multiple locally optimal solution.

Solution

False. When the features are linearly separable, we can push loss unregularized loss arbitrarily close to 0 but the the loss is still convex. The sum of two convex functions is convex, so there will be global optimum.

2. Let $\hat{\mathbf{w}} = \arg\min_{\mathbf{w}} J(\mathbf{w})$ be a global optimum. $\hat{\mathbf{w}}$ is typically sparse.

   Solution

   False. This may be true with l_1 regularization, but is not usually the case with l_2 regularization. If one considers the dual Lagrangian problem, $\hat{\mathbf{w}}$ will lie on some hypersphere at a point which will not generally have 0 values.

3. If the training data is linearly separable, then some weights w_j might become infinite if $\lambda = 0$.

   Solution

   True. By making the weights larger and larger we can push the loss to be arbitrarily close to 0 if $\lambda = 0$.

4. $l(\hat{\mathbf{w}}, \mathcal{D}_{\text{train}})$ always increases as we increase λ.

   Solution

   True. If one thinks in term of the Lagrangian dual problem increasing λ is contraining the weights further away from the global optimum by restricting them to a smaller hypersphere.

5. $l(\hat{\mathbf{w}}, \mathcal{D}_{\text{test}})$ always increases as we increase λ.

   Solution

   False. While the training loss may increase, we could be overfitting. Thus, sometimes increasing λ may decrease test loss.

Multi-class Logistic Regression

In multi-class logistic regression, suppose $Y \in \{y_1, \ldots, y_R\}$. A simplified version (with no bias term) is as follows. When $k < R$ the posterior probability is given by:

$$P(Y = y_k \mid X) = \frac{\exp(\langle w_k, X\rangle)}{1 + \sum_{j=1}^{R-1} \exp(\langle w_j, X\rangle)}. \tag{5}$$

For $k = R$, the posterior is

$$P(Y = y_R \mid X) = \frac{1}{1 + \sum_{j=1}^{R-1} \exp(\langle w_j, X\rangle)}. \tag{6}$$

To simplify notation, we can define $w_R = \mathbf{0}$ as a vector of all 0s. This gives us Equation 5 for all k.

1. How many parameters do we need to estimate? What are these parameters?

   Solution

   Assume the data is D-dimensional. Our parameters are the weights w_k each which is a D-dimensional vector. There are $(R-1)D$ parameters to estimate.

2.

3.

4.