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):

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

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. $z_2 = \sin(2\pi z_1)$
3. $z_3 = \exp(2w_2)$
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 AutoDiff (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

In the forward mode, we sequentially compute the values z_t and derivates of z_t with respect to w_1 in order for $t = 1, 2, \ldots, 6$.

Suppose we want to calculate $\frac{df}{dw_1}(u, v)$. Fix $w_1 = u$ and $w_2 = v$.

1. Compute $z_1 = u/v$.
2. Compute $\frac{dz_1}{w_1} = 1/v$.
3. Compute $z_2 = \sin(2\pi z_1)$.
4. Compute $\dfrac{dz_2}{dw_1} = \dfrac{\partial z_2}{\partial z_1}\dfrac{dz_1}{w_1} = 2\pi\cos(2\pi z_1)\dfrac{dz_1}{w_1}$.
5. Compute $z_3 = \exp(2v)$.
6. Compute $\dfrac{dz_3}{dw_1} = 0$.
7. Compute $z_4 = 3z_1 - z_3$.
8. Compute $\dfrac{dz_4}{dw_1} = \dfrac{\partial z_4}{\partial z_1}\dfrac{dz_1}{dw_1} + \dfrac{\partial z_4}{\partial z_3}\dfrac{dz_3}{dw_1} = 3\dfrac{dz_1}{dw_1} - \dfrac{dz_3}{dw_1}$.
9. Compute $z_5 = z_2 + z_4$.
10. Compute $\dfrac{dz_5}{dw_1} = \dfrac{\partial z_5}{\partial z_2}\dfrac{dz_2}{dw_1} + \dfrac{\partial z_5}{\partial z_4}\dfrac{dz_4}{dw_1} = \dfrac{dz_2}{dw_1} + \dfrac{dz_4}{dw_1}$.
11. Compute $z_6 = z_4 z_5$.
12. Compute $\dfrac{dz_6}{dw_1} = \dfrac{\partial z_6}{\partial z_4}\dfrac{dz_4}{dw_1} + \dfrac{\partial z_6}{\partial z_5}\dfrac{dz_5}{dw_1} = z_5\dfrac{dz_4}{dw_1} + z_4\dfrac{dz_5}{dw_1}$.

Each step can be computed by substituting values from the previous steps. The output is $\boxed{\dfrac{df}{dw_1}(u, v) = \dfrac{dz_6}{dw_1}.}$

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 with the assumption that you have evaluated the trace and already stored all the z_ts in memory already.

Solution

In the reverse mode, we compute $\frac{df}{dz_t} = \frac{dz_6}{dz_t}$ in order $t = 6, 5, \ldots, 0$. The general algorithm is

$$\frac{dz_6}{dz_t} = \sum_{c \text{ is a child of } t} \frac{dz_6}{dz_c}\frac{\partial z_c}{\partial z_t}. \tag{2}$$

1. Seed $\dfrac{dz_6}{dz_6} = 1$.
2. Compute $\dfrac{dz_6}{dz_5} = \dfrac{dz_6}{dz_6}\dfrac{\partial z_6}{\partial z_5} = z_4$.
3. Compute $\dfrac{dz_6}{dz_4} = \dfrac{dz_6}{dz_5}\dfrac{\partial z_5}{\partial z_4} + \dfrac{dz_6}{dz_6}\dfrac{\partial z_6}{\partial z_4} = \dfrac{dz_6}{dz_5} + \dfrac{dz_6}{dz_6}z_5 = z_4 + z_5$.
4. Compute $\dfrac{dz_6}{dz_3} = \dfrac{dz_6}{dz_4}\dfrac{\partial z_4}{\partial z_3} = -\dfrac{dz_6}{dz_4} = -z_4 - z_5$.
5. Compute $\dfrac{dz_6}{dz_2} = \dfrac{dz_6}{dz_5}\dfrac{\partial z_5}{\partial z_2} = \dfrac{dz_6}{dz_5} = z_4$.
6. Compute $\dfrac{dz_6}{dz_1} = \dfrac{dz_6}{dz_2}\dfrac{\partial z_2}{\partial z_1} + \dfrac{dz_6}{dz_4}\dfrac{\partial z_4}{\partial z_1} = 2\pi\cos(2\pi z_1)\dfrac{dz_6}{dz_2} + 3\dfrac{dz_6}{dz_4}$.
7. Compute $\dfrac{dz_6}{dw_1} = \dfrac{dz_6}{dz_1}\dfrac{\partial z_2}{\partial w_1} = \dfrac{1}{w_2}\dfrac{dz_6}{dz_1}$.
8. Compute $\dfrac{dz_6}{dw_2} = \dfrac{dz_6}{dz_1}\dfrac{\partial z_1}{\partial w_2} + \dfrac{dz_6}{dz_3}\dfrac{\partial z_3}{\partial w_2} = -\dfrac{w_1}{w_2^2}\dfrac{dz_6}{dz_1} + 2\exp(2w_2)\dfrac{dz_6}{dz_3}$.

Each step can be computed by substituting the output of one of the previous steps. From that, we obtain our desired result

$$\boxed{\frac{df}{dw} = \begin{pmatrix} \frac{dz_6}{dw_1} \\ \frac{dz_6}{dw_2} \end{pmatrix}.} \tag{3}$$

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. Define *and* plot a one dimensional, real valued function which is not continuous.

Solution

Consider the function

$$f(x) = \begin{cases} 2x, -1 & x < 0; \\ 0, & x = 0; \\ 2x + 1, & x > 0. \end{cases} \tag{4}$$

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

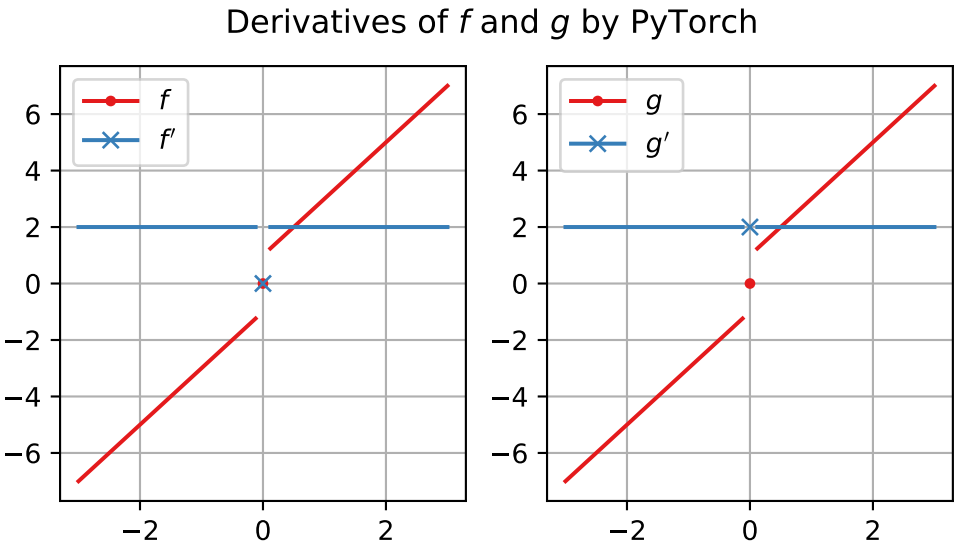

Figure 1: The functions from Listings 1 and 2 are plotted along with their PyTorch-computed derivatives.

2. Now write out your function in PyTorch. You should be able to define this function so that PyTorch returns a derivative of 0 at some point x_0.

Solution

Equation 4 is implemented as f in Listing 1. In Figure 1, you can see that PyTorch calculates the derivative as 0 at $x_0 = 0$ (left plot, blue `x`).

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

Listing 1: Equation 4 defined with the sign function factored out.

3. Now find another way to write out your function in PyTorch; do this so that it is exactly the same function. Do this in a way so that PyTorch now returns a derivative of 2 at exactly the same point x_0 that you obtained a derivative of 0 in the previous question.

Solution

Equation 4 is implemented as g in Listing 2. In Figure 1, you can see that PyTorch calculates the derivative as 2 at $x_0 = 0$ (right plot, blue `x`).

```
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 4 defined element-wise by recursing into the tensor.

4. There is a no sane definition of the derivative for your function. Yet, you should not only found a way to get PyTorch to provide a derivative, you should have also found a way to give you two *different* derivatives at exactly the same point (on the same function). What went wrong?

Solution

PyTorch is just keeping track of the operations and does a pointwise calculation. It's not paying attention to any local behvaior like continuity. In Listing 2, by changing the `else` arm, we can have it return any arbitrary number for the derivative. For instance, having it return `-3*x` would have PyTorch calculating the derivative as -3 at $x_0 = 0$.

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, differentiable line $f(x) = 2x$ in two ways: (1) verbosely using the sign function as f and (2) the canonical way as 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.

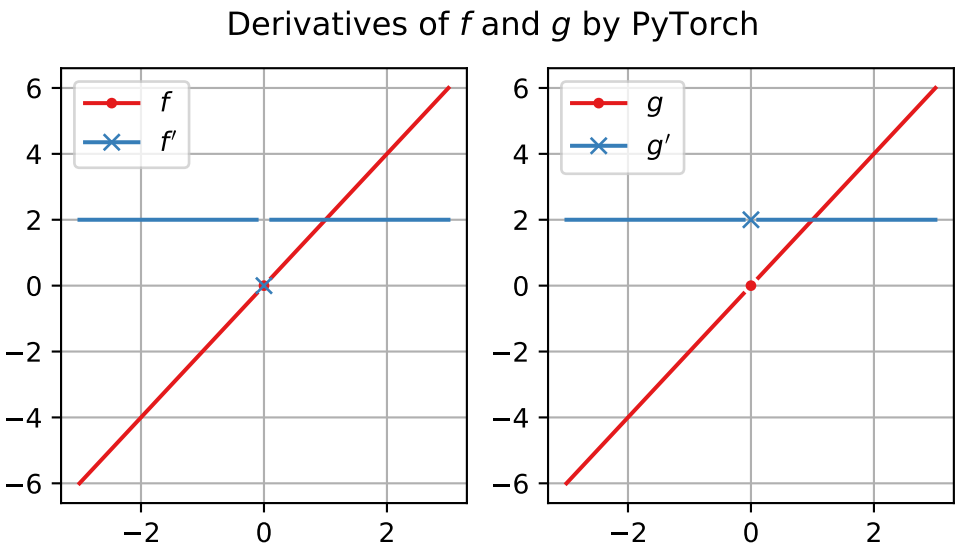

Figure 2: f and g from Listing 3 plotted along with their PyTorch-computed derivatives.

Problem 5: Elementary properties of l_2 regularized logisitic regression

The binary case

Consider minimizing

$$J(\mathbf{w}) = -l\left(\mathbf{w}, \mathcal{D}_{\text{train}}\right) + \lambda \left\|\mathbf{w}\right\|_2^2, \tag{5}$$

where

$$l\left(\mathbf{w}, \mathcal{D}\right) = \sum_j \log \mathbf{P}\left(y^j \mid \mathbf{x}^j, \mathbf{w}\right) \tag{6}$$

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:

$$P(Y = y_k \mid X) = \frac{\exp(\langle w_k, X \rangle)}{1 + \sum_{j=1}^{R-1} \exp(\langle w_j, X \rangle)}. \tag{7}$$

For $k = R$, the posterior is

$$P(Y = y_R \mid X) = \frac{1}{1 + \sum_{j=1}^{R-1} \exp(\langle w_j, X \rangle)}. \tag{8}$$

To simplify notation, we can define $w_R = \mathbf{0}$ as a vector of all 0s. This gives us Equation 7 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 $\boxed{(R-1)D}$ parameters to estimate.

2. Given N training samples $\left\{\left(x^1, y^1\right), \left(x^2, y^2\right), \ldots, \left(x^N, y^N\right)\right\}$, write down explicitly the log-likelihood function and simplify it as much as you can:

$$L\left(w_1, \ldots, w_{R-1}\right) = \sum_{j=1}^{N} \log\left(P\left(y^j \mid x^j, w\right)\right). \tag{9}$$

Solution

Let $y^j = y_{l^j}$, that is, let l^j be the class label of observation j. If we use the posterior probability in Equation 7, then, we have that

$$\log\left(P\left(y^j \mid x^j, w\right)\right) = \left\langle w_{l^j}, x^j \right\rangle - \log\left(1 + \sum_{k=1}^{R-1} \exp\left(\left\langle w_k, x^j \right\rangle\right)\right). \tag{10}$$

Substituting Equation 10 into Equation 9, we have

$$L\left(w_1, \ldots, w_{R-1}\right) = \sum_{j=1}^{N}\left(\left\langle w_{l^j}, x^j \right\rangle - \log\left(1 + \sum_{k=1}^{R-1} \exp\left(\left\langle w_k, x^j \right\rangle\right)\right)\right). \tag{11}$$

3. Compute the gradient of L with respect to each w_k and simplify it.

Solution

w_k is a D-dimensional vector so, $\frac{\partial L}{\partial w_k}$ will also be D-dimensional. Denote the features for the observations with class label k by $\mathcal{X}_k = \left\{x^j : l^j = k\right\}$.

We have that for $k = 1, 2, \ldots, R-1$:

$$\begin{aligned} \frac{\partial L}{\partial w_k} &= \sum_{x \in \mathcal{X}_k} x - \sum_{j=1}^{N} x^j \frac{\exp\left(\left\langle w_k, x^j \right\rangle\right)}{1 + \sum_{m=1}^{R-1} \exp\left(\left\langle w_m, x^j \right\rangle\right)} \\ &= \sum_{x \in \mathcal{X}_k} x - \sum_{j=1}^{N} x^j P\left(Y = y_k \mid X = x^j\right). \end{aligned} \tag{12}$$

4. Now add the regularization term λ and define a new objective function:

$$L\left(w_1, \ldots, w_{R-1}\right) = \sum_{j=1}^{N} \log\left(P\left(y^j \mid x^j, w\right)\right) + \frac{\lambda}{2} \sum_{l=1}^{R-1} \left\|w_l\right\|_2^2. \tag{13}$$

Compute the gradient of this new L with respect to each w_k.

Solution

We can just differentiate term by term. The gradeint of the first term comes from Equation 12. We can write $\|w_l\|_2^2 = w_{l1}^2 + w_{l2}^2 + \cdots + w_{lD}^2$, so we have that

$$\frac{\partial L}{\partial w_l} = \sum_{x \in \mathcal{X}_l} x - \sum_{j=1}^{N} x^j \frac{\exp\left(\langle w_l, x^j \rangle\right)}{1 + \sum_{m=1}^{R-1} \exp\left(\langle w_m, x^j \rangle\right)} + \lambda w_l. \tag{14}$$