Homework 1: Backpropagation

CSCI-GA 2572 Deep Learning

Spring 2021

The goal of homework 1 is to help you understand how to update network parameters by the using backpropagation algorithm.

For part 1, you need to answer the questions with mathematics equations. You should put all your answers in a PDF file and we will not accept any scanned hand-written answers. It is recommended to use LaTeX.

For part 2, you need to program with Python. It requires you to implement your own forward and backward pass without using autograd. You need to submit your `mlp.py` file for this part.

The due date of homework 1 is `23:55` of `02/25`. Submit the following files in a zip file `your_net_id.zip` through NYU classes:

- `theory.pdf`
- `mlp.py`

The following behaviors will result in penalty of your final score:

1. 5% penalty for submitting your files without using the correct format. (including naming the zip file, PDF file or python file wrong, or adding extra files in the zip folder, like the testing scripts from part 2).
2. 20% penalty for late submission within the first 24 hours. We will not accept any late submission after the first 24 hours.
3. 20% penalty for code submission that cannot be executed using the steps we mentioned in part 2. So please test your code before submit it.

1 Theory (50pt)

To answer questions in this part, you need some basic knowledge of linear algebra and matrix calculus. Also, you need to follow the instructions:

1. Every vector is treated as column vector.

2. You need to use the numerator-layout notation for matrix calculus. Please refer to Wikipedia about the notation.

3. You are only allowed to use vector and matrix. You cannot use tensor in any of your answer.

4. Missing transpose are considered as wrong answer.

1.1 Two-Layer Neural Nets

You are given the following neural net architecture:

$$\texttt{Linear}_1 \to f \to \texttt{Linear}_2 \to g$$

where $\texttt{Linear}_i(x) = \boldsymbol{W}^{(i)}\boldsymbol{x} + \boldsymbol{b}^{(i)}$ is the i-th affine transformation, and f, g are element-wise nonlinear activation functions. When an input $\boldsymbol{x} \in \mathbb{R}^n$ is fed to the network, $\hat{\boldsymbol{y}} \in \mathbb{R}^K$ is obtained as the output.

1.2 Regression Task

We would like to perform regression task. We choose $f(\cdot) = (\cdot)^+ = \texttt{ReLU}(\cdot)$ and g to be the identity function. To train this network, we choose MSE loss function $\ell_{\mathrm{MSE}}(\hat{\boldsymbol{y}}, \boldsymbol{y}) = \|\hat{\boldsymbol{y}} - \boldsymbol{y}\|^2$, where y is the target output.

(a) (1pt) Name and mathematically describe the 5 programming steps you would take to train this model with `PyTorch` using SGD on a single batch of data.

(1) forward pass (2) loss computation (3) clear gradients (4) compute gradients (backward pass) (5) update parameters (step)

(b) (5pt) For a single data point (x, y), write down all inputs and outputs for forward pass of each layer. You can only use variable $\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{W}^{(1)}, \boldsymbol{b}^{(1)}, \boldsymbol{W}^{(2)}, \boldsymbol{b}^{(2)}$ in your answer. (note that $\texttt{Linear}_i(\boldsymbol{x}) = \boldsymbol{W}^{(i)}\boldsymbol{x} + \boldsymbol{b}^{(i)}$).

Layer	Input	Output
$\texttt{Linear}_1$	x	$W^{(1)}x + b^{(1)}$
f	$W^{(1)}x + b^{(1)}$	$(W^{(1)}x + b^{(1)})^+$
$\texttt{Linear}_2$	$(Wx + b)^+$	$W^{(2)}(W^{(1)}x + b^{(1)})^+ + b^{(2)}$
g	$W^{(2)}(W^{(1)}x + b^{(1)})^+ + b^{(2)}$	$W^{(2)}(W^{(1)}x + b^{(1)})^+ + b^{(2)}$
Loss	$W^{(2)}(W^{(1)}x + b^{(1)})^+ + b^{(2)}$, y	$\|W^{(2)}(W^{(1)}x + b^{(1)})^+ + b^{(2)} - y\|^2$

(c) (8pt) Write down the gradient calculated from the backward pass. You can only use the following variables: $\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{W}^{(1)}, \boldsymbol{b}^{(1)}, \boldsymbol{W}^{(2)}, \boldsymbol{b}^{(2)}, \frac{\partial \ell}{\partial \hat{\boldsymbol{y}}}, \frac{\partial \boldsymbol{z}_2}{\partial \boldsymbol{z}_1}, \frac{\partial \hat{\boldsymbol{y}}}{\partial \boldsymbol{z}_3}$ in your answer, where $\boldsymbol{z}_1, \boldsymbol{z}_2, \boldsymbol{z}_3, \hat{\boldsymbol{y}}$ are the outputs of $\texttt{Linear}_1, f, \texttt{Linear}_2, g$.

First, let's figure out the dimension of each vector, matrix, tensor:

We already have $x \in \mathbb{R}^n$ and $z_3, \hat{y} \in \mathbb{R}^K$, so let's assume $z_1, z_2 \in \mathbb{R}^a$, then we have

$$\begin{aligned} W^{(1)} \in \mathbb{R}^{a\times n}, &\quad b^{(1)} \in \mathbb{R}^a \\ W^{(2)} \in \mathbb{R}^{K\times a}, &\quad b^{(2)} \in \mathbb{R}^K \end{aligned}$$

Using the numerator-layout notation for the matrix calculus we have:

The dimension of all the gradients are

$$\begin{aligned} \frac{\partial \ell}{\partial W^{(1)}} \in \mathbb{R}^{n\times a}, &\quad \frac{\partial \ell}{\partial b^{(1)}} \in \mathbb{R}^{1\times a} \\ \frac{\partial \ell}{\partial W^{(2)}} \in \mathbb{R}^{a\times K}, &\quad \frac{\partial \ell}{\partial b^{(2)}} \in \mathbb{R}^{1\times K} \end{aligned}$$

and

$$\begin{aligned} &\frac{\partial \ell}{\partial \hat{y}} \in \mathbb{R}^{1\times K}, \quad \frac{\partial \hat{y}}{\partial z_3} \in \mathbb{R}^{K\times K}, \quad \frac{\partial z_3}{\partial z_2} \in \mathbb{R}^{K\times a}, \quad \frac{\partial z_2}{\partial z_1} \in \mathbb{R}^{a\times a} \\ &\frac{\partial z_1}{\partial b^{(1)}} \in \mathbb{R}^{a\times a}, \quad \frac{\partial z_1}{\partial W^{(1)}} \in \mathbb{R}^{a\times n\times a} \\ &\frac{\partial z_3}{\partial b^{(2)}} \in \mathbb{R}^{K\times K}, \quad \frac{\partial z_3}{\partial W^{(2)}} \in \mathbb{R}^{K\times a\times K} \end{aligned}$$

It is easy to see that

$$\begin{aligned} \frac{\partial \ell}{\partial z_3} &= \frac{\partial \ell}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial z_3} \\ \frac{\partial \ell}{\partial z_1} &= \frac{\partial \ell}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial z_3} \frac{\partial z_3}{\partial z_2} \frac{\partial z_2}{\partial z_1} = \frac{\partial \ell}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial z_3} W^{(2)} \frac{\partial z_2}{\partial z_1} \end{aligned}$$

For bias, we have

$$\begin{aligned} \frac{\partial \ell}{\partial b^{(1)}} &= \frac{\partial \ell}{\partial z_1} \frac{\partial z_1}{\partial b^{(1)}} = \frac{\partial \ell}{\partial z_1} \\ \frac{\partial \ell}{\partial b^{(2)}} &= \frac{\partial \ell}{\partial z_3} \frac{\partial z_3}{\partial b^{(2)}} = \frac{\partial \ell}{\partial z_3} \end{aligned}$$

For weight, since $\frac{\partial z_1}{\partial W^{(1)}}$ and $\frac{\partial z_3}{\partial W^{(2)}}$ are tensors, so we need to do an extra step:

$$\frac{\partial \ell}{\partial z_1} = \begin{bmatrix} \frac{\partial \ell}{\partial (z_1)_1} & \frac{\partial \ell}{\partial (z_1)_2} & \cdots & \frac{\partial \ell}{\partial (z_1)_a} \end{bmatrix}$$

$$\frac{\partial z_1}{\partial W^{(1)}} = \begin{bmatrix} \frac{\partial (z_1)_1}{\partial W^{(1)}} \\ \frac{\partial (z_1)_2}{\partial W^{(1)}} \\ \vdots \\ \frac{\partial (z_1)_a}{\partial W^{(1)}} \end{bmatrix}$$

$$\frac{\partial \ell}{\partial z_3} = \begin{bmatrix} \frac{\partial \ell}{\partial (z_3)_1} & \frac{\partial \ell}{\partial (z_3)_2} & \cdots & \frac{\partial \ell}{\partial (z_1)_K} \end{bmatrix}$$

$$\frac{\partial z_3}{\partial W^{(2)}} = \begin{bmatrix} \frac{\partial (z_3)_1}{\partial W^{(2)}} \\ \frac{\partial (z_3)_2}{\partial W^{(2)}} \\ \vdots \\ \frac{\partial (z_3)_K}{\partial W^{(2)}} \end{bmatrix}$$

Then we have

$$\frac{\partial \ell}{\partial W^{(1)}} = \frac{\partial \ell}{\partial z_1}\frac{\partial z_1}{\partial W^{(1)}} = \sum_i \frac{\partial \ell}{\partial (z_1)_i}\frac{\partial (z_1)_i}{\partial W^{(1)}} = x\frac{\partial \ell}{\partial z_1}$$

$$\frac{\partial \ell}{\partial W^{(2)}} = \frac{\partial \ell}{\partial z_3}\frac{\partial z_3}{\partial W^{(2)}} = \sum_i \frac{\partial \ell}{\partial (z_3)_i}\frac{\partial (z_3)_i}{\partial W^{(2)}} = z_2\frac{\partial \ell}{\partial z_3}$$

So the answer is

Parameter	Gradient
$W^{(1)}$	$x\frac{\partial \ell}{\partial \hat{y}}\frac{\partial \hat{y}}{\partial z_3}W^{(2)}\frac{\partial z_2}{\partial z_1}$
$b^{(1)}$	$\frac{\partial \ell}{\partial \hat{y}}\frac{\partial \hat{y}}{\partial z_3}W^{(2)}\frac{\partial z_2}{\partial z_1}$
$W^{(2)}$	$(W^{(1)}x + b^{(1)})^{+}\frac{\partial \ell}{\partial \hat{y}}\frac{\partial \hat{y}}{\partial z_3}$
$b^{(2)}$	$\frac{\partial \ell}{\partial \hat{y}}\frac{\partial \hat{y}}{\partial z_3}$

(d) (3pt) Show us the elements of $\frac{\partial \boldsymbol{z_2}}{\partial \boldsymbol{z_1}}$, $\frac{\partial \hat{\boldsymbol{y}}}{\partial \boldsymbol{z_3}}$ and $\frac{\partial \ell}{\partial \hat{\boldsymbol{y}}}$ (be careful about the dimensionality)?

$\frac{\partial z_2}{\partial z_1}$ and $\frac{\partial \hat{y}}{\partial z_3}$ are matrices and $\frac{\partial \ell}{\partial \hat{y}}$ is row vector.

All of the off diagonal elements of $\frac{\partial z_2}{\partial z_1}$ and $\frac{\partial \hat{y}}{\partial z_3}$ are 0.

For $\frac{\partial z_2}{\partial z_1}$, the diagonal elements are

$$\left(\frac{\partial z_2}{\partial z_1}\right)_{ii} = \mathbf{1}_{[(z_1)_i > 0]}$$

For $\frac{\partial \hat{y}}{\partial z_3}$, the diagonal elements are

$$\left(\frac{\partial \hat{y}}{\partial z_3}\right)_{ii} = 1$$

For $\frac{\partial \hat{y}}{\partial z_3}$, the elements are

$$\left(\frac{\partial \ell}{\partial \hat{y}}\right)_{i} = 2(\hat{y} - y)_i$$

1.3 Classification Task

We would like to perform multi-class classification task, so we set both $f, g = \sigma$, the logistic sigmoid function $\sigma(z) \doteq (1 + \exp(-z))^{-1}$.

(a) (5pt + 8pt + 3pt) If you want to train this network, what do you need to change in the equations of (b), (c) and (d), assuming we are using the same MSE loss function.

Layer	Input	Output
Linear$_1$	x	$W^{(1)}x + b^{(1)}$
f	$W^{(1)}x + b^{(1)}$	$\sigma(W^{(1)}x + b^{(1)})$
Linear$_2$	$\sigma(W^{(1)}x + b^{(1)})$	$W^{(2)}\sigma(W^{(1)}x + b^{(1)}) + b^{(2)}$
g	$W^{(2)}\sigma(W^{(1)}x + b^{(1)}) + b^{(2)}$	$\sigma(W^{(2)}\sigma(W^{(1)}x + b^{(1)}) + b^{(2)})$
Loss	$\sigma(W^{(2)}\sigma(W^{(1)}x + b^{(1)}) + b^{(2)}), y$	$\|\sigma(W^{(2)}\sigma(W^{(1)}x + b^{(1)}) + b^{(2)}) - y\|^2$

Parameter	Gradient
$W^{(1)}$	$x \frac{\partial \ell}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial z_3} W^{(2)} \frac{\partial z_2}{\partial z_1}$
$b^{(1)}$	$\frac{\partial \ell}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial z_3} W^{(2)} \frac{\partial z_2}{\partial z_1}$
$W^{(2)}$	$\sigma(W^{(1)}x + b^{(1)}) \frac{\partial \ell}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial z_3}$
$b^{(2)}$	$\frac{\partial \ell}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial z_3}$

All of the off diagonal elements of $\frac{\partial z_2}{\partial z_1}$ and $\frac{\partial \hat{y}}{\partial z_3}$ are 0.

$$\left(\frac{\partial z_2}{\partial z_1}\right)_{ii} = \sigma((z_1)_i)(1 - \sigma((z_1)_i))$$

$$\left(\frac{\partial \hat{y}}{\partial z_3}\right)_{ii} = \sigma((z_3)_i)(1 - \sigma((z_3)_i))$$

$$\left(\frac{\partial \ell}{\partial \hat{y}}\right)_i = 2(\hat{y} - y)_i$$

(b) (5pt + 8pt + 3pt) Now you think you can do a better job by using a *Binary Cross Entropy* (BCE) loss function $\ell_{\mathrm{BCE}}(\hat{\boldsymbol{y}}, \boldsymbol{y}) = \frac{1}{K}\sum_{i=1}^{K} -\left[y_i \log(\hat{y}_i) + (1-y_i)\log(1-\hat{y}_i)\right]$. What do you need to change in the equations of (b), (c) and (d)?

Layer	Input	Output
Linear$_1$	x	$W^{(1)}x + b^{(1)}$
f	$W^{(1)}x + b^{(1)}$	$\sigma(W^{(1)}x + b^{(1)})$
Linear$_2$	$\sigma(W^{(1)}x + b^{(1)})$	$W^{(2)}\sigma(W^{(1)}x + b^{(1)}) + b^{(2)}$
g	$W^{(2)}\sigma(W^{(1)}x + b^{(1)}) + b^{(2)}$	$\sigma(W^{(2)}\sigma(W^{(1)}x + b^{(1)}) + b^{(2)})$
Loss	$\sigma(W^{(2)}\sigma(W^{(1)}x + b^{(1)}) + b^{(2)}), y$	$-\frac{1}{K}\left[y^T \log(\sigma(W^{(2)}\sigma(W^{(1)}x + b^{(1)}) + b^{(2)})) + (1-y)^T \log(1 - \sigma(W^{(2)}\sigma(W^{(1)}x + b^{(1)}) + b^{(2)}))\right]$

Parameter	Gradient
$W^{(1)}$	$x\frac{\partial \ell}{\partial \hat{y}}\frac{\partial \hat{y}}{\partial z_3}W^{(2)}\frac{\partial z_2}{\partial z_1}$
$b^{(1)}$	$\frac{\partial \ell}{\partial \hat{y}}\frac{\partial \hat{y}}{\partial z_3}W^{(2)}\frac{\partial z_2}{\partial z_1}$
$W^{(2)}$	$\sigma(W^{(1)}x + b^{(1)})\frac{\partial \ell}{\partial \hat{y}}\frac{\partial \hat{y}}{\partial z_3}$
$b^{(2)}$	$\frac{\partial \ell}{\partial \hat{y}}\frac{\partial \hat{y}}{\partial z_3}$

All of the off diagonal elements of $\frac{\partial z_2}{\partial z_1}$ and $\frac{\partial \hat{y}}{\partial z_3}$ are 0.

$$\left(\frac{\partial z_2}{\partial z_1}\right)_{ii} = \sigma((z_1)_i)(1 - \sigma((z_1)_i))$$

$$\left(\frac{\partial \hat{y}}{\partial z_3}\right)_{ii} = \sigma((z_3)_i)(1 - \sigma((z_3)_i))$$

$$\left(\frac{\partial \ell}{\partial \hat{y}}\right)_i = \frac{1}{K}\frac{y_i - \hat{y}_i}{\hat{y}_i - \hat{y}_i^2}$$

(c) (1pt) Things are getting better. You realize that not all intermediate hidden activations need to be binary (or soft version of binary). You decide to use $f(\cdot) = (\cdot)^+$ but keep g as σ. Explain why this choice of f can be beneficial for training a (deeper) network.

no vanishing gradient

2 Implementation (50pt)

You need to implement the forward pass and backward pass for `Linear`, `ReLU`, `Sigmoid`, `MSE loss`, and `BCE loss` in the attached `mlp.py` file. We provide three example test cases `test1.py`, `test2.py`, `test3.py`. We will test your implementation with other hidden test cases, so please create your own test cases to make sure your implementation is correct.

Extra instructions:

1. Please use Python version ≥ 3.7 and PyTorch version 1.7.1. We recommend you to use Miniconda the manage your virtual environment.
2. We will put your `mlp.py` file under the same directory of the hidden test scripts and use the command `python hiddenTestScriptName.py` to check your implementation. So please make sure the file name is `mlp.py` and it can be executed with the example test scripts we provided.
3. You are not allowed to use PyTorch autograd functionality in your implementation.
4. Be careful about the dimensionality of the vector and matrix in PyTorch. It is not necessarily follow the the Math you got from part 1.