Homework 1 : backpropagation

Marie

January 27, 2023

0.1 Regression task

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.

- *Set up*: define and instantiate the network class, get a batch from the data loader
- *Predict output for the batch* (forward pass) : compute the output for each module of the network, eg for a linear layer the output is $W * X + b$ where W is the weight matrix, X is the input vector and b is bias. For a non-linear transformation . The output is a vector (number of dimensions can be tuned).
- Evaluate*loss* based on the difference between the prediction and labels. Different functions can be used as the cost function eg `nn.BCEloss()` for binary classification tasks (where $l_n = -w_n * [y_n \cdot logx_n + (1 - y_n) \cdot log(1 - x_n)]$ is average across N samples in batch). For SGD, the loss function is evaluated for only one data point (introduces some randomness in gradient descent butoverall faster than training on the whole batch, especially if there is some redundancy between samples)
- Evaluate *gradient* and update the weights in the direction of maximum change: *backpropagation* is automatically carried out by `autograd()` which computes the product of the Jacobian matrix and the input vector. Mathematically, backpropagation depends on the *chain rule*. The weights are updated based on the learning rate η : $w \leftarrow w - \eta \frac{\partial L(s,w)}{\partial w}$
- repeat for n epochs

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

Layer	Input	Output
$Linear_1$	x	$W^{(1)} * x + b^{(1)}$
f	$W^{(1)} * x + b$	$\begin{cases} W^{(1)} * x + b, & \text{if} W^{(1)} * x + b > 0 \\ 0, & \text{otherwise} \end{cases}$
$Linear_2$	f output	$W^{(2)} * (W^{(1)} * (x) + b^{(1)}) + b^{(2)}$
g	$W^{(2)} * (W^{(1)} * (x) + b^{(1)}) + b^{(2)}$	$W^{(2)} * (W^{(1)} * (x) + b^{(1)}) + b^{(2)}$
Loss	$Linear_2$ output	$\|\|W^{(2)} * (W^{(1)} * (x) + b^{(1)}) + b^{(2)} - y\|\|^2$

Write down the gradient calculated from the backward pass. You can only use the following variables: $x, y, W_{(1)}, b_{(1)}, W_{(2)}, b_{(2)}, \frac{\partial l}{\partial \hat{y}}, \frac{\partial z_2}{\partial z_1}, \frac{\partial \hat{y}}{\partial z_3}$ in your answer where $z_1, z_2, z_3, \hat{y}$ are the outputs of $Linear_1, f, Linear_2, g$.

Working backwards along the graph :
$l(y, \hat{y}) = \frac{1}{N} \sum_{i=1}^{N} (\hat{y_i} - y_i)^2$

$$\frac{\partial l}{\partial \hat{y}} = \frac{1}{N}\sum_{i=1}^{N} 2 * (\hat{y}_i - y_i)$$
$$\frac{\partial l}{\partial \hat{y}} = \frac{2N}{N}\sum_{i=1}^{N} (\hat{y}_i - y_i)$$
$$\frac{\partial l}{\partial \hat{y}} = 2 * \sum_{i=1}^{N} (\hat{y}_i - y_i)$$

$$\hat{y} = \begin{cases} z_3 & \text{if} z_3 > 0 \\ 0, & \text{otherwise} \end{cases}$$

$$z_3 = W_2 * z_2 + b_2$$

$$\frac{\partial l}{\partial W_2} = \frac{\partial l}{\partial z_3} \cdot \frac{\partial z_3}{\partial W_2}$$
$$\frac{\partial l}{\partial W_2} = \frac{\partial l}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial z_3} \cdot X^T$$

$$\frac{\partial l}{\partial b_2} = \frac{\partial l}{\partial z_3} \cdot \frac{\partial z_3}{\partial b_2}$$
$$\frac{\partial l}{\partial b_2} = \frac{\partial l}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial z_3} * 1$$

$$\frac{\partial l}{\partial W_1} = \frac{\partial l}{\partial z_1} \cdot \frac{\partial z_1}{\partial w_1} \quad \frac{\partial l}{\partial W_1} = \frac{\partial l}{\partial z_2} \cdot \frac{\partial z_2}{\partial z_1} \cdot X^T$$

where : $\frac{\partial z_2}{\partial z_1} = \begin{cases} z_1 & \text{if } z_1 > 0 \\ 0, & \text{otherwise} \end{cases}$

and: $\frac{\partial l}{\partial z_2} = \frac{\partial l}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial z_3} \cdot \frac{\partial z_3}{\partial z_2}$
$$\frac{\partial l}{\partial z_2} = \frac{\partial l}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial z_3} \cdot W_2$$

$$\frac{\partial l}{\partial b_1} = \frac{\partial l}{\partial z_1} \cdot \frac{\partial z_1}{\partial b_1}$$
$$\frac{\partial l}{\partial b_1} = \frac{\partial l}{\partial z_1} * 1$$

Parameters	Gradient
$W_{(1)}$	$\frac{\partial l}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial z_3} \cdot W_2 \cdot \frac{\partial z_2}{\partial z_1} \cdot X^T$
$b_{(1)}$	$\frac{\partial l}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial z_3} \cdot W_2 \cdot \frac{\partial z_2}{\partial z_1}$
$W_{(2)}$	$\frac{\partial l}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial z_3} \cdot X^T$
$b_{(2)}$	$\frac{\partial l}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial z_3}$

Show us the elements of $\frac{\partial z_2}{\partial z_1}$, $\frac{\partial \hat{y}}{\partial z_3}$ and $\frac{\partial l}{/partial\hat{y}}$

$\frac{\partial z_2}{\partial z_1} = \begin{cases} z_1 & \text{if } z_1 > 0 \\ 0, & \text{otherwise} \end{cases}$ where z_2 and z_1 are vectors

$\frac{\partial l}{\partial \hat{y}} = -2(\hat{y} - y)$ where y and $\hat{y}$ are vectors

$$\hat{y} = \begin{cases} z_3 & \text{if} z_3 > 0 \\ 0, & \text{otherwise} \end{cases}$$

0.2 Classification task

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

In question (b), the outputs of f and g will change.

In question (c), the formula for the gradients going through the activation function will change. With ReLU, the gradient was 1 in the input was positive, 0 otherwise. With the sigmoid function we have :

$\hat{y} = (1 + exp(-z_3))^-1$ $\frac{\partial \hat{y}}{\partial z_3} = -1 * (-1 * exp(-z_3)) * (1 + exp(-z_3))^-2$

$\frac{\partial \hat{y}}{\partial z_3} = exp(-z_3) * (1 + exp(-z_3))^-2$
$\frac{\partial \hat{y}}{\partial z_3} = exp(-z_3) * \frac{1}{(1+exp(-z_3))^2}$
$\frac{\partial \hat{y}}{\partial z_3} = \frac{1}{1+exp(-z_3)} * \frac{exp(-z_3)+1-1}{(1+exp(-z_3)} \frac{\partial \hat{y}}{\partial z_3} = \frac{1}{1+exp(-z_3)} * (1 - \frac{1}{(1+exp(-z_3)}$

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

We now have $\frac{\partial l}{\partial \hat{y}} = \frac{1}{K} \sum_{i=1}^{K} -[\frac{y_i}{\hat{y}_i} + \frac{y_i - 1}{1 - \hat{y}_i}]$

Things are getting better. You realize that not all intermediate hidden ac- tivations 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.
Benefits of using ReLU include :

1. *Sparsity* : sets negative gradients to zero
2. On positive numbers, ReLU is *equivariant to scale* : if multiply input by 2, output is multiplied by 2 (unlike eg sigmoid or tanh which saturate)
3. Simple *computation* allows for faster training?