Deep learning HW 1

wbg231

September 2023

1 theroy

1.1 two layer neural networks

1.1.1 Regression task

(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

- here I am assuming that we have already defined the model.
- Step one: Randomly select one sample from the training set S. Call it $(x^i, y^i) \in S$
- step two: Compute the output of the network $\tilde{y}^i = f(x^i, W^1, W^2, b^1, b^2)$
- step three: find the loss for that example $L(\tilde{y}^i, y^i)$
- step four: compute the network gradients using backpropagation that is find $\frac{\partial L}{\partial W^1}, \frac{\partial L}{\partial W^2}$
- step five: update the network parameters using the gradients and the learning rate η that is $W^1 = W^1 - \eta \frac{\partial L}{\partial W^1}, W^2 = W^2 - \eta \frac{\partial L}{\partial W^2}$

(b) (4pt)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 $Linear_i(x) = W^{(i)}x + b^{(i)}$).

- our input x^i is passed to our first linear layer to produce $s^1 = W^{(1)}x^i + b^{(1)}$
- $s^{(1)}$ is then passed to our first activation function f to produce $a^{(1)} = f(s^{(1)}) = 5reLU(s^1)$
- $a^{(1)}$ is then passed to our second linear layer to produce $s^{(2)} = W^{(2)}a^{(1)} + b^{(2)}$
- $s^{(2)}$ is then passed to our second activation function g to produce our final prediction $\tilde{y}^{(i)} = I(s^{(2)}) = s^{(2)}$
- our prediction $\tilde{y}^{(i)}$ is finally compared against the true value of example i $y^{(i)}$ to produce our loss $L(\tilde{y}^{(i)}, y^{(i)}) = ||\tilde{y}^{(i)} - y^{(i)}||_2$

- and those our the steps of this networks forward pass.

(c) write down the gradients from the backwards pass.

- first we want to find $\frac{\partial C}{\partial \tilde{y}}$ we know that $C \in \mathbb{R}$ and $\tilde{y} \in \mathbb{R}^K$ thus we will have $\frac{\partial C}{\partial \tilde{y}} \in \mathbb{R}^{1\times K}$
- then more formally we can see that $\frac{\partial C}{\tilde{y}} = \frac{\partial}{\partial \tilde{y}}||y-\tilde{y}||_2 = \frac{\partial}{\partial \tilde{y}}(y-\tilde{y})^t(y-\tilde{y}) = -2(y-\tilde{y})^t$
- now we want to find $\frac{\partial \tilde{y}}{\partial s_2} = \frac{\partial s^2}{s^2} = I \in \mathbb{R}^{K\times K}$
- now we want to find $\frac{\partial s^2}{\partial a^1} = \frac{\partial}{\partial a^1}(W^{(2)}a^{(1)} + b^{(2)}) = W^{(2)} \in \mathbb{R}^{K\times D}$
- we also want $\frac{\partial s^2}{W^2} = \frac{\partial}{\partial W^2}(W^{(2)}a^{(1)}+b^{(2)}) = \left(\left(a^{1^t}0^t,\cdots\right),\left(0^t,a^{1^t},0^t,\cdots\right),\cdots\left(0^t,\cdots,a^{1^t}\right)\right) \in \mathbb{R}^{K\times(K\times D)}$
- finally we can see that $\frac{\partial s^2}{b^2} = \frac{\partial}{\partial b^2}(W^{(2)}a^{(1)} + b^{(2)}) = I \in \mathbb{R}^{K\times K}$
- going further back we want $\frac{\partial a^1}{\partial s^1} = \frac{\partial}{\partial s^1}(5ReLU)(s^1) = \frac{\partial}{\partial s^1}\begin{cases} 5s_i^1 & \text{if } s_i^1 > 0 \\ 0 & \text{if } s_i^1 < 0\end{cases} = \begin{cases} 5 & \text{if } s_i^1 > 0 \\ 0 & \text{if } s_i^1 < 0\end{cases} \in \mathbb{R}^{D\times D}$ so then we have $\frac{\partial a^1}{\partial s^1}_{i,j} = \begin{cases} 5 & \text{if } s_i^1 > 0 \text{and} i = j \\ 0 & \text{if } s_i^1 < 0 \text{ or } i \neq j\end{cases}$
- we also want $\frac{\partial s^1}{\partial W^1} = \frac{\partial}{\partial W^1}(W^{(1)}x^{(i)}+b^{(1)}) = \left(\left(x^{i^t}0^t,\cdots\right),\left(0^t,x^{i^t},0^t,\cdots\right),\cdots\left(0^t,\cdots,x^{i^t}\right)\right) \in \mathbb{R}^{D\times(D\times N)}$
- finally we can see that $\frac{\partial s^1}{\partial b^1} = \frac{\partial}{\partial b^1}(W^{(1)}x^{(i)} + b^{(1)}) = I \in \mathbb{R}^{D\times D}$
- and those are all the gradients for this network.

(d) show the elements of $\frac{\partial a_1}{s_1}, \frac{\tilde{y}}{s_2}, \frac{\partial C}{\partial \tilde{y}}$

- suppose that we know that $a_1 \in \mathbb{R}^D$ and $s_1 \in \mathbb{R}^D$ then we can see that $\frac{\partial a_1}{s_1} \in \mathbb{R}^{D\times D}$ and $\frac{\partial s^1}{\partial W^1} = \frac{\partial}{\partial W^1}(W^{(1)}x^{(i)}+b^{(1)}) = \left(\left(x^{i^t}0^t,\cdots\right),\left(0^t,x^{i^t},0^t,\cdots\right),\cdots\left(0^t,\cdots,x^{i^t}\right)\right) \in \mathbb{R}^{D\times(D\times N)}$ meaning that $\frac{\partial a^1}{\partial s^1}_{i,j} = \begin{cases} 5 & \text{if } s_i^1 > 0 \text{and} i = j \\ 0 & \text{if } s_i^1 < 0 \text{ or } i \neq j\end{cases}$
- now suppose that we know that $\tilde{y} \in \mathbb{R}^K$ and $s_2 \in \mathbb{R}^K$ then we can see that $\frac{\partial \tilde{y}}{s_2} \in \mathbb{R}^{K\times K}$ and $\frac{\partial \tilde{y}}{\partial s_2} = \frac{\partial s^2}{s^2} = I \in \mathbb{R}^{K\times K}$ thus we can see that $\frac{\partial \tilde{y}}{s_2}_{i,j} = \begin{cases} 1 & \text{if } i = j \\ 0 & \text{if } i \neq j\end{cases}$
- finally we can see that $C \in \mathbb{R}$ and $\tilde{y} \in \mathbb{R}^k$ thus $\frac{\partial C}{\partial \tilde{y}} \in \mathbb{R}^{1\times K}$ and $\frac{\partial C}{\tilde{y}} = \frac{\partial}{\partial \tilde{y}}||y-\tilde{y}||_2 = \frac{\partial}{\partial \tilde{y}}(y-\tilde{y})^t(y-\tilde{y}) = -2(y-\tilde{y})^t$ thus we can see that $\frac{\partial C}{\partial \tilde{y}}_{i,j} = -2(y-\tilde{y})_i$

1.1.2 Classification task

(a) 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 squared Euclidean distance loss function

- we need to change our forward pass in the following ways.
  - set our first layer activation function to be $f(s^1) = tanh(s^1)$
  - set our second layer activation function to be $\sigma(s^2) = \frac{1}{1+e^{-s^2}}$
- we also need to change our backward pass in the following ways.
  - change $\frac{\partial a^1}{\partial s^1} = \frac{\partial f}{\partial s^1} = \frac{\partial}{\partial s^1} tanh(s^1) = \frac{\partial}{\partial s^1} \frac{sinh(s^1)}{coshs^1} = 1 - tanh^2(s^1)$
  - also change $\frac{\partial \tilde{y}}{\partial s^2} = \frac{\partial}{\partial s^2}(\sigma(s^2)) = \frac{\partial}{\partial s^2}(\frac{1}{1+e^{-s^2}}) = \frac{e^{-s^2}}{((1+e^{-s^2})^2)} = \sigma(s^2)(1-\sigma(s^2))$
- the following things change in the elements of our gradients
  - we have $a^1 \in \mathbb{R}^D, s^1 \in \mathbb{R}^d$ $\frac{\partial a^1}{\partial s^1} \in \mathbb{R}^{d \times d}$ where $\frac{\partial a^1}{\partial s^1}_{i,j} = \begin{cases} 1 - tanh^2(s_i^1) & \text{if } i = j \\ 0 & \text{if } i \neq j \end{cases}$
  - similarly we have $\tilde{y} \in \mathbb{R}^k, s^2 \in \mathbb{R}^k$ meaning that $\frac{\partial \tilde{y}}{\partial s^2} \in \mathbb{R}^{k \times k}$ and $\frac{\partial \tilde{y}}{\partial s^2}_{i,j} = \begin{cases} \sigma(s_i^2)(1-\sigma(s_i^2)) & \text{if } i = j \\ 0 & \text{if } i \neq j \end{cases}$

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

- in our forward pass we need to make the following change
  - set $C(y, \tilde{y}) = \frac{1}{K}\sum_{i=1}^{k} -[y_i log(\tilde{y}_i) + (1 - y_i) log(1 - \tilde{y}_i)]$
- in the backwards pass we need to make the following change
  - set $\frac{\partial C}{\partial \tilde{y}} = \frac{\partial}{\partial \tilde{y}} \frac{1}{K}\sum_{i=1}^{k} -[y_i log(\tilde{y}_i) + (1 - y_i) log(1 - \tilde{y}_i)] = -\frac{1}{k}(N_1 \sum_{y_i=1} \frac{1}{\tilde{y}} - (K - N_1)\sum_{y_i \neq 1} \frac{1}{1-\tilde{y}})$ where N_1 is the number of samples of class one.
- finally the elements of our gradients can be changed as
  - $C \in \mathbb{R}, \tilde{y} \in \mathbb{R}^K, \frac{\partial \tilde{y}}{\partial \tilde{y}} \in \mathbb{R}^{1 \times K}$ and we also have $\frac{\partial C}{\partial \tilde{y}}_i = \begin{cases} -\frac{1}{k}\frac{N_1}{\tilde{y}_i} & \text{if } y_i = 1 \\ \frac{1}{k}\frac{K-N_1}{1-\tilde{y}_i} & \text{if } y_i \neq 1 \end{cases}$
- could also just do this in vector form

(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 $f(*) = (*)^+$ but keep g as σ. Explain why this choice of f can be beneficial for training a (deeper) network.

- suppose we have a deep neural network, and two examples x_i, x_j such that we are quite confidante (say we know it 100%) in the Classification of x_i and only fairly confidante in the Classification of x_j say we are only %75 sure, in a binary or soft binary activation function these two examples will have a similar output that is $s^1(x_j) \approx s^1(x_i)$, then these s^1 values are passed onto the second layer of the network with little to no information about how confidante our previous layer was. You can intuitively understand how over many layers this could lead our networks activations to become distorted

1.2 Conceptual Questions

(a) why is the softmax function really softmaxarg function

- the soft max function is defined as $\sigma(x_i) = \frac{e^{x_i}}{\sum_i e^{x^i}}$ so intuitively what this function is doing is normalizing our our network outputs be all positive and sum to one. in this way we can think of them as probabilities.
- thus if we are using the softmax in a Classification we will predict the class with the highest probability ie (assuming index class labels) $\tilde{y} = argmax_i \sigma(x_i)$