EECS 189 Introduction to Machine Learning
Fall 2020

DIS7

This discussion was released **Friday, October 16**.

This discussion serves as an introduction to neural networks. You will through a simple example of backpropagation and build intuition for the ReLU nonlinearity and gradient descent training process through visualizations in the Jupyter **[notebook]**. This notebook takes a long time to do the initial network training! You should begin the training process then return to the theory part of this discussion while you wait.

1 ReLU SGD Visualization

Work through the **[notebook]** to explore how a simple network with ReLU non-linearities adapts to model a function with SGD updates. Training the networks takes 5-10 minutes depending on whether you run locally or on datahub and the server load, so you should start the training process (run through the train all layers cell) then return to the theory part of the discussion while training occurs.

As you walk through the notebook, pay attention to how the slopes and elbows of the ReLU functions change during training and how they impact the shape of the final learned function.

2 Backpropagation

In this problem, we will explore the chain rule of differentiation, and provide some algorithmic motivation for the backpropagation algorithm. Those of you who have taken CS170 may recognize a particular style of algorithmic thinking that underlies the computation of gradients.

Let us begin by working with simple functions of two variables.

(a) Define the functions $f(x) = x^2$ and $g(x) = x$, and $h(x_1, x_2) = x_1^2 + x_2^2$. Compute the derivative of $\ell(x) = h(f(x), g(x))$ with respect to x.

Solution: We can write $\ell(x) = x^4 + x^2$, and from there, this is just a simple differentiation problem, and we have $\ell'(x) = 4x^3 + 2x$.

It is also helpful to think of this from the chain rule perspective, where we have $\ell(x) = f(x)^2 + g(x)^2$, and differentiating using the chain rule yields

$$\begin{aligned}\ell'(x) &= 2f(x)f'(x) + 2g(x)g'(x)\\ &= 2x^2(2x) + 2(x)\cdot 1\\ &= 4x^3 + 2x.\end{aligned}$$

(b) Chain rule of multiple variables: Assume that you have a function given by $f(x_1, x_2, \ldots, x_n)$, and that $g_i(w) = x_i$ for a scalar variable w. How would you compute $\frac{\mathrm{d}}{\mathrm{d}w} f(g_1(w), g_2(w), \ldots, g_n(w))$? What is its computation graph?

Solution: This is the chain rule for multiple variables. In general, we have

$$\frac{\mathrm{d}f}{\mathrm{d}w} = \sum_{i=1}^{n} \frac{\partial f}{\partial x_i} \frac{\partial x_i}{\partial w} = \frac{\partial f}{\partial \mathbf{x}} \frac{\partial \mathbf{x}}{\partial w}.$$

In the literature of neural networks, you will find the people often abuse the notation in which they replace the total derivative $\frac{\mathrm{d}f}{\mathrm{d}w}$ with the partial derivative symbol $\frac{\partial f}{\partial w}$. *This is **NOT** correct mathematically* but it simplifies the notation as we can see in the following part.

The function graph of this computation is given in Figure 1.

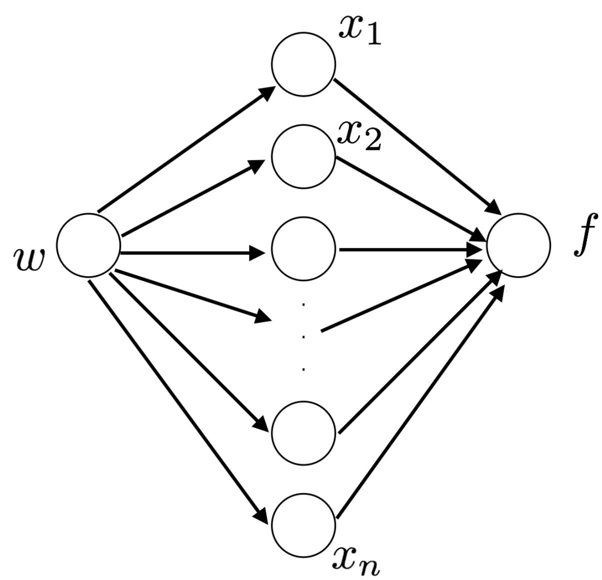

Figure 1: Example function computation graph

(c) Let $\mathbf{w}_1, \mathbf{w}_2, \ldots, \mathbf{w}_n \in \mathbb{R}^d$, and we refer to these variables together as $\mathbf{W} \in \mathbb{R}^{n \times d}$. We also have $\mathbf{x} \in \mathbb{R}^d$ and $y \in \mathbb{R}$. Consider the function

$$f(\mathbf{W}, \mathbf{x}, y) = \left(y - \sum_{i=1}^{n} \phi(\mathbf{w}_i^\top \mathbf{x} + \mathbf{b}_i) \right)^2.$$

Write out the function computation graph (also sometimes referred to as a pictorial representation of the network). This is a directed graph of decomposed function computations, with the function at one end (which we will call the sink), and the variables $\mathbf{W}, \mathbf{x}, y$ at the other end (which we will call the sources).

Solution:

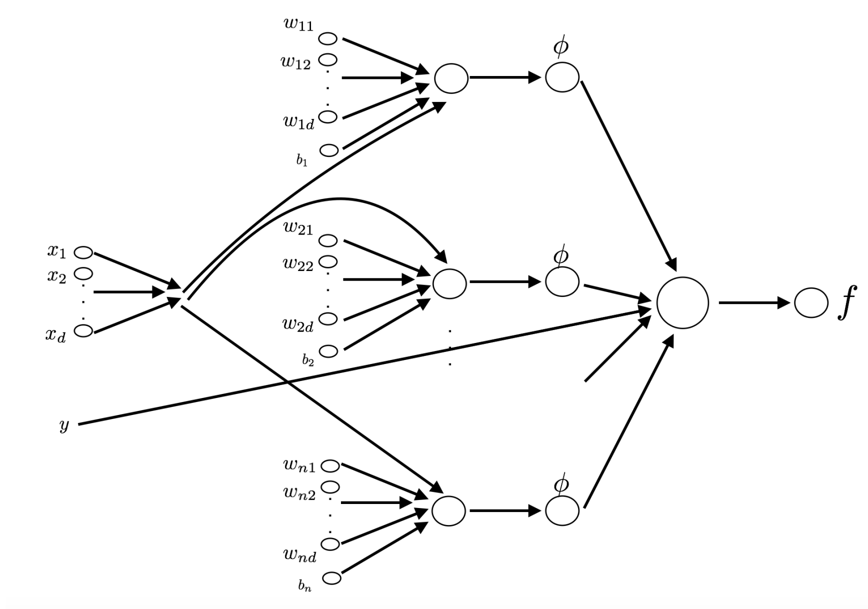

Figure 2: Example function computation graph

(d) Define the cost function

$$\ell(\mathbf{x}) = \frac{1}{2}\|\mathbf{W}^{(2)}\mathbf{\Phi}\left(\mathbf{W}^{(1)}\mathbf{x} + \mathbf{b}\right) - \mathbf{y}\|_2^2, \tag{1}$$

where $\mathbf{W}^{(1)} \in \mathbb{R}^{d\times d}$, $\mathbf{W}^{(2)} \in \mathbb{R}^{d\times d}$, and $\mathbf{\Phi} : \mathbb{R}^d \to \mathbb{R}^d$ is some nonlinear transformation. **Compute the partial derivatives $\frac{\partial \ell}{\partial \mathbf{x}}, \frac{\partial \ell}{\partial \mathbf{W}^{(1)}}, \frac{\partial \ell}{\partial \mathbf{W}^{(2)}}$, and $\frac{\partial \ell}{\partial \mathbf{b}}$. Track the dimensions of each element of the derivatives as you go and make sure they make sense.**

In order to keep track of the partial derivatives, it is incredibly helpful to define intermediate variables. We suggest using the ones below, but you are free to define your own.

$$\begin{aligned}
\mathbf{x}^{(1)} &= \mathbf{W}^{(1)}\mathbf{x} + \mathbf{b} \\
\mathbf{x}^{(2)} &= \mathbf{\Phi}(\mathbf{x}^{(1)}) \\
\mathbf{x}^{(3)} &= \mathbf{W}^{(2)}\mathbf{x}^{(2)} \\
\mathbf{x}^{(4)} &= \mathbf{x}^{(3)} - \mathbf{y} \\
\ell &= \frac{1}{2}\|\mathbf{x}^{(4)}\|_2^2.
\end{aligned}$$

Remember that the superscripts represent the index rather than the power operator.

Solution: Using the provided intermediate variables, we have

$$\begin{aligned}
\frac{\partial \ell}{\partial \mathbf{x}^{(4)}} &= \mathbf{x}^{(4)\top} \\
\frac{\partial \ell}{\partial \mathbf{x}^{(3)}} &= \frac{\partial \ell}{\partial \mathbf{x}^{(4)}}\frac{\partial \mathbf{x}^{(4)}}{\partial \mathbf{x}^{(3)}} = \frac{\partial \ell}{\partial \mathbf{x}^{(4)}} = \mathbf{x}^{(4)\top} \\
\frac{\partial \ell}{\partial \mathbf{x}^{(2)}} &= \frac{\partial \ell}{\partial \mathbf{x}^{(3)}}\frac{\partial \mathbf{x}^{(3)}}{\partial \mathbf{x}^{(2)}} = \frac{\partial \ell}{\partial \mathbf{x}^{(3)}}\mathbf{W}^{(2)} = \mathbf{x}^{(4)\top}\mathbf{W}^{(2)}
\end{aligned}$$

$$
\begin{aligned}
\frac{\partial \ell}{\partial \mathbf{W}^{(2)}} &= \frac{\partial \ell}{\partial \mathbf{x}^{(3)}} \frac{\partial \mathbf{x}^{(3)}}{\partial \mathbf{W}^{(2)}} = \mathbf{x}^{(2)} \frac{\partial \ell}{\partial \mathbf{x}^{(3)}} = \mathbf{x}^{(2)} \mathbf{x}^{(4)\top} \\
\frac{\partial \ell}{\partial \mathbf{x}^{(1)}} &= \frac{\partial \ell}{\partial \mathbf{x}^{(2)}} \frac{\partial \mathbf{\Phi}}{\partial \mathbf{x}^{(1)}} = \mathbf{x}^{(4)\top} \mathbf{W}^{(2)} \frac{\partial \mathbf{\Phi}}{\partial \mathbf{x}^{(1)}} \\
\frac{\partial \ell}{\partial \mathbf{x}} &= \frac{\partial \ell}{\partial \mathbf{x}^{(1)}} \frac{\partial \mathbf{x}^{(1)}}{\partial \mathbf{x}} = \frac{\partial \ell}{\partial \mathbf{x}^{(1)}} \mathbf{W}^{(1)} = \mathbf{x}^{(4)\top} \mathbf{W}^{(2)} \frac{\partial \mathbf{\Phi}}{\partial \mathbf{x}^{(1)}} \mathbf{W}^{(1)} \\
\frac{\partial \ell}{\partial \mathbf{b}} &= \frac{\partial \ell}{\partial \mathbf{x}^{(1)}} \frac{\partial \mathbf{x}^{(1)}}{\partial \mathbf{b}} = \frac{\partial \ell}{\partial \mathbf{x}^{(1)}} \\
\frac{\partial \ell}{\partial \mathbf{W}^{(1)}} &= \frac{\partial \ell}{\partial \mathbf{x}^{(1)}} \frac{\partial \mathbf{x}^{(1)}}{\partial \mathbf{W}^{(1)}} = \mathbf{x} \frac{\partial \ell}{\partial \mathbf{x}^{(1)}} = \mathbf{x} \mathbf{x}^{(4)\top} \mathbf{W}^{(2)} \frac{\partial \mathbf{\Phi}}{\partial \mathbf{x}^{(1)}}.
\end{aligned}
$$

The easy trick to solve these derivatives is to "guess an expression" so that the dimension is correct on both sides. Notice that we abuse the notation. These derivatives should be total rather than partial.

A more formal way to solve these requires doing the expansion. For example, $\frac{\partial \ell}{\partial \mathbf{W}^{(1)}} = \frac{\partial \ell}{\partial \mathbf{x}^{(1)}} \frac{\partial \mathbf{x}^{(1)}}{\partial \mathbf{W}^{(1)}}$. However, the right hand side is a 3D tensor, for which the matrix codebook does not provide us useful information. We need to do that manually. Notice that $x_k^{(1)} = \sum_l W_{kl}^{(1)} x_l + b_k$, so we have

$$
\begin{aligned}
\frac{\partial \ell}{\partial W_{ij}^{(1)}} &= \sum_k \frac{\partial \ell}{\partial x_k^{(1)}} \frac{\partial x_k^{(1)}}{\partial W_{ij}^{(1)}} \\
&= \sum_k \sum_l \frac{\partial \ell}{\partial x_k^{(1)}} \left(\epsilon_{ik} \epsilon_{jl} x_l \right) \\
&= \frac{\partial \ell}{\partial x_i^{(1)}} x_j
\end{aligned}
$$

so that

$$
\frac{\partial \ell}{\partial \mathbf{W}^{(1)}} = \frac{\partial \ell}{\partial \mathbf{x}^{(1)}} \frac{\partial \mathbf{x}^{(1)}}{\partial \mathbf{W}^{(1)}} = \mathbf{x} \frac{\partial \ell}{\partial \mathbf{x}^{(1)}}. \tag{2}
$$

(e) Compare the computation complexity of computing the $\frac{\partial \ell}{\partial \mathbf{W}}$ for Equation (1) using the analytic derivatives and numerical derivatives. Remember that if we want to compute the derivative of some function $f(x)$ at $x = 3$, we can use

$$
\frac{d}{dx} f(x)|_{x=3} = \lim_{\epsilon \to 0} \frac{f(3 + \epsilon) - f(3)}{\epsilon}
$$

Solution: For numerical differentiation, what we do is to use the following first order formula

$$
\frac{\partial \ell}{\partial W_{ij}} = \frac{\ell\left(W_{ij} + \epsilon, \cdot\right) - \ell\left(W_{ij}, \cdot\right)}{\epsilon}.
$$

We need $O(d^4)$ operations in order to compute $\frac{\partial \ell}{\partial \mathbf{W}}$. On the other hand, it only takes $O(d^2)$ operations to compute it analytically.

(f) What is the intuitive interpretation of taking a partial derivative of the output with respect to a particular node of this function graph?

Solution: The partial derivative measures how much the function changes when that particular variable is changed, keeping all the other variables in our computation fixed. In particular, assume in the previous example that we are computing $\frac{\partial f}{\partial u_i}$ for some node of the graph u_i. In order to see what this looks like, we fix the values of all the parameters of the network that do not influence u_i, and vary u_i by a small amount. The partial derivative then tells us the rate at which the function changes. In order to move downhill, we then see that we must toggle u_i in the direction that decreases the function value.

(g) Discuss how gradient descent would work on the function $f(\mathbf{W}, \mathbf{x}, y)$ if we use backpropagation as a subroutine to compute gradients with respect to the parameters $\mathbf{W}$ (with $\mathbf{x}$ and y given).

Solution: In order to compute a gradient update, we require $w_t = w_{t-1} - \gamma \nabla f(w_t)$. The gradient step can be computed by backpropagation above. However, we also require the function value in order to compute each of the gradients, and note that as w changes, the function values at each of the nodes changes. Therefore, the gradient descent algorithm proceeds with a forward pass (in order to compute the function values at each of the nodes) followed by a backward pass via backpropagation.

Contributors:

- Anant Sahai
- Ashwin Pananjady
- Josh Sanz
- Yichao Zhou