Recitation 5
Neural Networks

10-301/10-601: Introduction to Machine Learning

2022-03-02

1 Matrix Calculus

Consider $\mathbf{x} \in \mathbb{R}^p$, $\mathbf{y} \in \mathbb{R}^r$, $\mathbf{z} \in \mathbb{R}^n$ where $\mathbf{z} = g(\mathbf{y})$, and $\mathbf{y} = f(\mathbf{x})$. We want to derive $d\mathbf{z}/d\mathbf{x}$ (a vector form of the scalar chain rule).

1. If $\mathbf{x}$, $\mathbf{y}$, and $\mathbf{z}$ were all scalars, what would dz/dx be?

   Shape matching:

2. Fill in the following shapes:

$$\frac{d\mathbf{y}}{d\mathbf{x}}: \qquad \frac{d\mathbf{z}}{d\mathbf{y}}: \qquad \frac{d\mathbf{z}}{d\mathbf{x}}:$$

3. Therefore, the correct derivative is

$$\frac{d\mathbf{z}}{d\mathbf{x}} =$$

   Generalizing a single element: The more rigorous method of computing such derivatives is by computing the scalar derivative for a single element, then generalizing this to all elements by turning the scalar form into a matrix form.

4. Fill in the single element:

$$\frac{dz_k}{dx_i} =$$

2 Forward Propagation

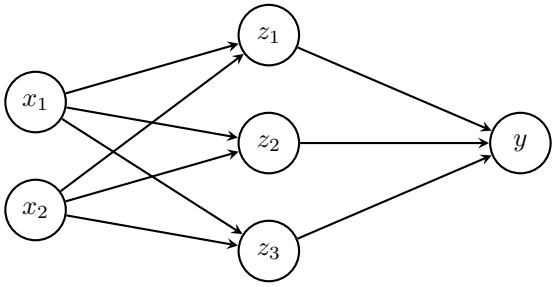

Figure 1: Neural Network For Example Questions

Forward Propagation is the process of calculating the value of your loss function, given data, weights and activation functions. Given the input data $\mathbf{x}$, we can transform it by the given weights, $\boldsymbol{\alpha}$, then apply the corresponding activation function to it and finally pass the result to the next layer. Forward propagation does not involve taking derivatives and proceeds from the input layer to the output layer.

Network Overview Consider the neural network with one hidden layer shown in Figure 2. The input layer consists of 2 features $\mathbf{x} = [x_1, x_2]^T$, the hidden layer has 3 nodes with output $\mathbf{z} = [z_1, z_2, z_3]^T$, and the output layer is a scalar $\hat{y}$. We also add a bias to the input, $x_0 = 1$ and the output of the hidden layer $z_0 = 1$, both of which are fixed to 1.

$\boldsymbol{\alpha}$ is the matrix of weights from the inputs to the hidden layer and $\boldsymbol{\beta}$ is the matrix of weights from the hidden layer to the output layer. $\alpha_{j,i}$ represents the weight going *to* the node z_j in the hidden layer *from* the node x_i in the input layer (e.g. $\alpha_{1,2}$ is the weight from x_2 to z_1), and $\boldsymbol{\beta}$ is defined similarly. We will use a **tanh** activation function for the hidden layer and no activation for the output layer.

Network Details Equivalently, we define each of the following.

The input:

$$\mathbf{x} = [x_0, x_1, x_2]^T \tag{1}$$

Linear combination at the first (hidden) layer:

$$a_j = \sum_{i=0}^{2} \alpha_{j,i} \cdot x_i, \ \forall j \in \{1, \ldots, 3\} \tag{2}$$

Activation at the first (hidden) layer:

$$z_j = \tanh(a_j) = \frac{e^{a_j} - e^{-a_j}}{e^{a_j} + e^{-a_j}}, \ \forall j \in \{1, \ldots, 3\} \tag{3}$$

$$\mathbf{z} = [z_0, z_1, z_2, z_3]^T \tag{4}$$

Linear combination at the second (output) layer:

$$\hat{y} = \sum_{j=0}^{3} \beta_j \cdot z_j, \tag{5}$$

Here we fold in the bias term $\alpha_{j,0}$ by thinking of $x_0 = 1$, and fold in β_0 by thinking of $z_0 = 1$.

Loss We will use Squared error loss, $\ell(\hat{y}, y)$:

$$\ell(\hat{y}, y) = \frac{1}{2}(\hat{y} - y)^2 \tag{6}$$

We initialize the network weights as:

$$\boldsymbol{\alpha} = \begin{bmatrix} 0 & 1 & 2 \\ 2 & 1 & 0 \\ 0 & 2 & 0 \end{bmatrix}$$

$$\boldsymbol{\beta} = \begin{bmatrix} 0 & 1 & 2 & 2 \end{bmatrix}$$

For the following questions, we use $y = 3$.

1. Why and how do we include a bias term in the input and in the hidden-layer?

2. Why do we need to use nonlinear activation functions in our neural net?

3. **Scalar Form:**

- Given $x_1 = 1$, $x_2 = 2$, What are the values of a?

$$a_1 = \sum_{i=0}^{2} \alpha_{1,i} x_i =$$
$$a_2 = \sum_{i=0}^{2} \alpha_{2,i} x_i =$$
$$a_3 = \sum_{i=0}^{2} \alpha_{3,i} x_i =$$

- Given $z_1 = 0, z_2 = 1, z_3 = 0$ calculate $\hat{y}, l$

4. **Vector Form:** Find the vector form of forward computation, given $\mathbf{x}$ is a column vector.

3 Backward Propagation

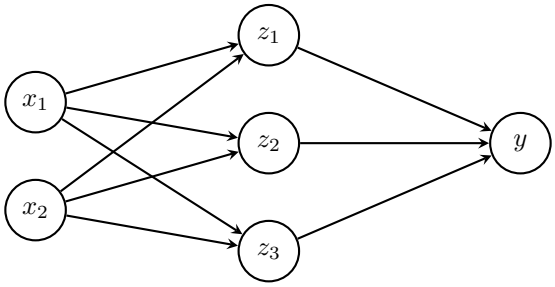

Figure 2: Neural Network For Example Questions (same as Figure 1)

Given a Neural Network and a corresponding loss function $J(\theta)$, backpropagation gives us the gradient of the loss function with respect to the weights of the neural network. The method is called *backward* propagation because we calculate the gradients of the final layer of weights first, then proceed backward to the first layer. In a simple neural network with one hidden layer, the partial derivatives that we need for learning are $\frac{\partial \ell}{\partial \alpha_{ij}}$ and $\frac{\partial \ell}{\partial \beta_{kj}}$, and we need to apply chain rule recursively to obtain these. Note that in implementation, it is easier to use matrix/vector forms to conduct computations.

1. Many gradients are calculated in back propagation. Which of these gradients are used to update the weights? Do not include intermediate value(s) used to calculate these gradient(s).

2. **Scalar Form:** Given $x_1 = 1$, $x_2 = 2$, what are the values of $\frac{\partial \ell}{\partial \beta_1}$, $\frac{\partial \ell}{\partial \alpha_{1,1}}$?

   Hint: $\frac{\partial \tanh(x)}{\partial x} = 1 - \tanh(x)^2$

Table 1: *tanh* values

x	1	2	3	4	5	6	7	8	9
$\tanh(x)$	0.76159	0.96403	0.99505	0.99933	0.99991	0.99999	0.99999	0.99999	0.99999

$$\frac{\partial \ell}{\partial \beta_1} = \frac{\partial \ell}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial \beta_1}$$

$$\frac{\partial \ell}{\partial \alpha_{1,1}} = \frac{\partial \ell}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial z_1} \frac{\partial z_1}{\partial a_1} \frac{\partial a_1}{\partial \alpha_{1,1}}$$

3. **Vector Form:** What are the values of $\frac{\partial \ell}{\partial \boldsymbol{\beta}}$, $\frac{\partial \ell}{\partial \boldsymbol{\alpha}}$?