Implementation from scratch a neural network

Nguyen Trung Hai

Nov 2019

Introduction

Let us implement from scratch the neural network as shown in Fig 1. Here we use three hidden layers but it can be easily extended to arbitrary number of hidden layers. The activation function of the hidden layers is sigmoid and that of output layer is softmax. The loss function is cross entropy.

Notations

- Let n denote the batch size of input data matrix.
- Let d, d_1, d_2 and k denote the dimension of input data, the number of units in the first, second hidden and the output layers, respectively.
- $\mathbf{X}$ is the input matrix with shape (n, d).
- $\mathbf{y}$ and $\hat{\mathbf{y}}$ are the true and predicted output matrices with shape (n, k). Each row is a one-hot encoded vector.
- $\mathbf{W_1}$ is the weight matrix connecting the input layer and the first hidden layer. Its shape is (d, d_1).
- $\mathbf{W_2}$ is the weight matrix connecting the first and the second hidden layers. Its shape is (d_1, d_2).
- $\mathbf{W_O}$ is the weight matrix connecting the second hidden layer and the output layer. Its shape is (d_2, k).
- $\mathbf{b}_1$, $\mathbf{b}_2$ and $\mathbf{b}_O$ are bias (row) vectors of the first, second hidden and the output layers, respectively. Their shapes are $(1, d_1)$, $(1, d_2)$ and $(1, k)$, respectively.

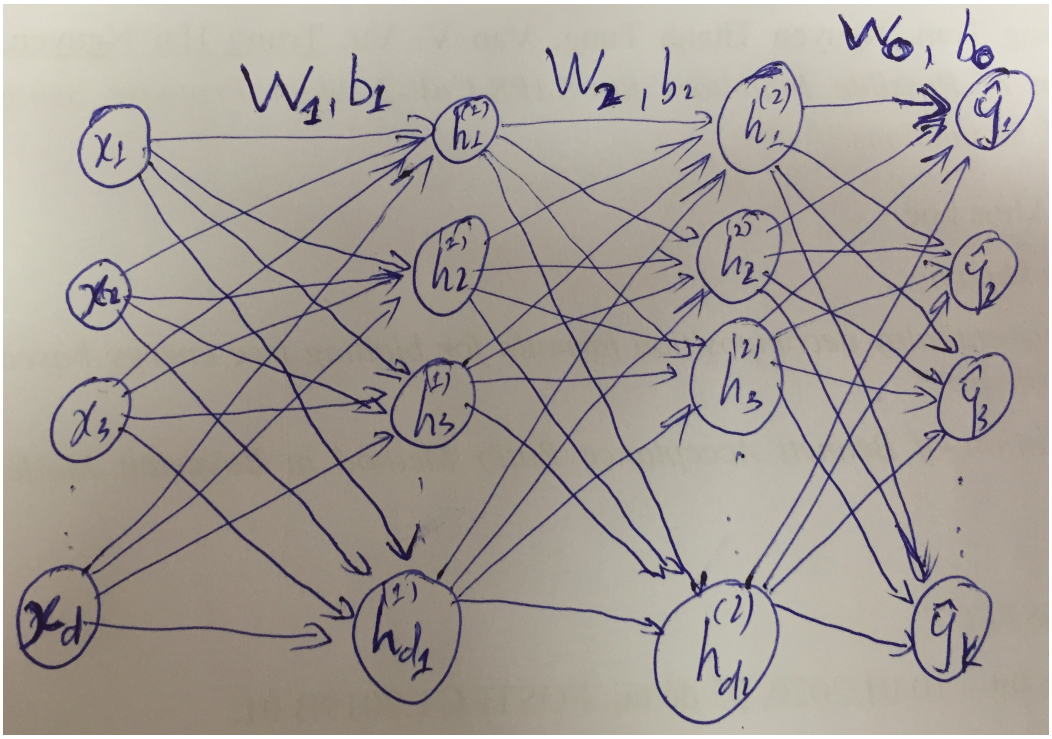

Figure 1: **A densely connected feedforward neural network.**

Feed forward equations

$$\mathbf{Z}_1 = \mathbf{X}\mathbf{W_1} + \mathbf{b}_1, \quad \mathbf{Z}_1 \text{ has shape } (n, d_1), \tag{1}$$

$$\mathbf{H}_1 = \sigma(\mathbf{Z}_1), \quad \mathbf{H}_1 \text{ has shape } (n, d_1), \tag{2}$$

$$\mathbf{Z}_2 = \mathbf{H}_1\mathbf{W_2} + \mathbf{b}_2, \quad \mathbf{Z}_2 \text{ has shape } (n, d_2), \tag{3}$$

$$\mathbf{H}_2 = \sigma(\mathbf{Z}_2), \quad \mathbf{H}_2 \text{ has shape } (n, d_2), \tag{4}$$

$$\mathbf{Z}_O = \mathbf{H}_2\mathbf{W_O} + \mathbf{b}_O, \quad \mathbf{Z}_O \text{ has shape } (n, k), \tag{5}$$

$$\hat{\mathbf{y}} = \text{softmax}(\mathbf{Z}_O), \quad \hat{\mathbf{y}} \text{ has shape } (n, k). \tag{6}$$

Note that the bias vectors are row vectors. Therefore, their addition with matrices assumes broadcasting along the first dimension.

Back propagation equations

Cross entropy loss function.

$$L = \text{CE}(\hat{\mathbf{y}}, \mathbf{y}) = \sum_{i=1}^{n}\sum_{j=1}^{d_U} -y_{i,j}\log\hat{y}_{i,j}. \tag{7}$$

$\frac{\partial L}{\partial \mathbf{W_O}}$ and $\frac{\partial L}{\partial \mathbf{b}_O}$

$$\delta_O \equiv \frac{\partial L}{\partial \mathbf{Z}_O} = \hat{\mathbf{y}} - \mathbf{y}, \quad \delta_O \text{ has shape } (n, k). \tag{8}$$

See https://web.stanford.edu/class/cs224n/readings/gradient-notes.pdf.

$$\begin{aligned} \frac{\partial L}{\partial \mathbf{W_O}} &= \frac{\partial L}{\partial \mathbf{Z}_O} \frac{\partial \mathbf{Z}_O}{\partial \mathbf{W_O}} \\ &= \mathbf{H}_2^T \times \delta_O \end{aligned} \tag{9}$$

Make sure shapes of matrices in the above equation are right: $(d_2, k) = (d_2, n) \times (n, k)$.

$$\begin{aligned} \frac{\partial L}{\partial \mathbf{b}_O} &= \frac{\partial L}{\partial \mathbf{Z}_O} \frac{\partial \mathbf{Z}_O}{\mathbf{b}_O} \\ &= \mathbb{1}_{1\times n} \times \delta_O. \end{aligned} \tag{10}$$

Make sure shapes of matrices in the equation above are right: $(1, k) = (1, n) \times (n, k)$.

$\frac{\partial L}{\partial \mathbf{W_2}}$ and $\frac{\partial L}{\partial \mathbf{b}_2}$

$$\begin{aligned} \delta_2 &= \frac{\partial L}{\partial \mathbf{Z}_2} \\ &= \frac{\partial L}{\partial \mathbf{Z}_O} \frac{\partial \mathbf{Z}_O}{\partial \mathbf{H}_2} \frac{\partial \mathbf{H}_2}{\partial \mathbf{Z}_2} \\ &= \delta_\mathbf{O} \times \mathbf{W_O}^T \odot \mathbf{H}_2 \odot (\mathbb{1} - \mathbf{H}_2), \end{aligned} \tag{11}$$

where $\odot$ denotes element-wise multiplication. The shape of δ_2 should be the same as that of $\mathbf{Z}_2$ which is (n, d_2). Let's make sure the shapes of matrices in the above equation are right: $(n, d_2) = (n, k) \times (k, d_2) \odot (n, d_2) \odot (n, d_2)$. Note that the derivative of the sigmoid function $\sigma(z)$ is

$$\frac{d\sigma(z)}{dz} = \sigma(z)(1 - \sigma(z)). \tag{12}$$

$$\begin{aligned} \frac{\partial L}{\partial \mathbf{W_2}} &= \frac{\partial L}{\partial \mathbf{Z}_2} \frac{\partial \mathbf{Z}_2}{\partial \mathbf{W_2}} \\ &= \mathbf{H}_1^T \times \delta_2. \end{aligned} \tag{13}$$

Let's make sure the shapes of matrices in the above equation are right: $(d_1, d_2) = (d_1, n) \times (n, d_2)$.

$$\begin{aligned} \frac{\partial L}{\partial \mathbf{b}_2} &= \frac{\partial L}{\partial \mathbf{Z}_2} \frac{\partial \mathbf{Z}_2}{\mathbf{b}_2} \\ &= \mathbb{1}_{1\times n} \times \delta_2. \end{aligned} \tag{14}$$

Let's make sure the shapes of matrices in the above equation are right: $(1, d_2) = (1, n) \times (n, d_2)$.

$\frac{\partial L}{\partial \mathbf{W_1}}$ and $\frac{\partial L}{\partial \mathbf{b}_1}$

$$\begin{aligned}\delta_1 &= \frac{\partial L}{\partial \mathbf{Z}_1} \\ &= \frac{\partial L}{\partial \mathbf{Z}_2}\frac{\partial \mathbf{Z}_2}{\partial \mathbf{H}_1}\frac{\partial \mathbf{H}_1}{\partial \mathbf{Z}_1} \\ &= \delta_{\mathbf{2}} \times {\mathbf{W_2}}^T \odot \mathbf{H}_1 \odot (\mathbb{1} - \mathbf{H}_1), \end{aligned} \tag{15}$$

The shape of δ_1 should be the same as that of $\mathbf{Z}_1$ which is (n, d_1). Let's make sure the shapes of matrices in the above equation are right: $(n, d_1) = (n, d_2) \times (d_2, d_1) \odot (n, d_1) \odot (n, d_1)$.

$$\begin{aligned}\frac{\partial L}{\partial \mathbf{W_1}} &= \frac{\partial L}{\partial \mathbf{Z}_1}\frac{\partial \mathbf{Z}_1}{\partial \mathbf{W_1}} \\ &= \mathbf{X}^T \times \delta_1. \end{aligned} \tag{16}$$

Let's make sure the shapes of matrices in the above equation are right: $(d, d_1) = (d, n) \times (n, d_1)$.

$$\begin{aligned}\frac{\partial L}{\partial \mathbf{b}_1} &= \frac{\partial L}{\partial \mathbf{Z}_1}\frac{\partial \mathbf{Z}_1}{\mathbf{b}_1} \\ &= \mathbb{1}_{1\times n} \times \delta_1. \end{aligned} \tag{17}$$

Let's make sure the shapes of matrices in the above equation are right: $(1, d_W) = (1, n) \times (n, d_W)$.

Gradient descent

- **Step 1:** Randomly initialize the weight matrices, $\mathbf{W_1}$, $\mathbf{W_2}$ and $\mathbf{W_O}$ and bias vectors $\mathbf{b}_1$, $\mathbf{b}_2$ and $\mathbf{b}_O$.
- **Step 2:** Use feed forward equations to compute $\mathbf{Z}_1$, $\mathbf{H}_1$, $\mathbf{Z}_2$, $\mathbf{H}_2$, $\mathbf{Z}_O$ and $\hat{\mathbf{y}}$.
- **Step 3:** Use back propagation equations to compute the derivatives $\frac{\partial L}{\partial \mathbf{W_O}}$, $\frac{\partial L}{\partial \mathbf{b}_O}$, $\frac{\partial L}{\partial \mathbf{W_2}}$, $\frac{\partial L}{\partial \mathbf{b_2}}$, $\frac{\partial L}{\partial \mathbf{W_1}}$ and $\frac{\partial L}{\partial \mathbf{b}_1}$.
- **Step 4:** Update the weight matrices and bias vectors. $\forall \mathbf{A} \in \{\mathbf{W_1}, \mathbf{W_2}, \mathbf{W_O}\}$ and $\forall \mathbf{b} \in \{\mathbf{b_1}, \mathbf{b_2}, \mathbf{b_O}\}$

$$\begin{aligned}\mathbf{A} &:= \mathbf{A} - \eta\frac{\partial L}{\partial \mathbf{A}}, \\ \mathbf{b} &:= \mathbf{b} - \eta\frac{\partial L}{\partial \mathbf{b}}, \end{aligned} \tag{18}$$

  where η is the learning rate.

- Repeat **Steps 2–4** for many iterations.

Note that in **Steps 2** and **3** we can use the whole matrix $\mathbf{X}$ (gradient descent), or just a few rows like 32, 64, 128 ... (mini-batch gradient descent) or even a single row (stochastic gradient descent).