Recitation 6
Neural Networks

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

10/15/2021

1 Forward Propagation Explained

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.

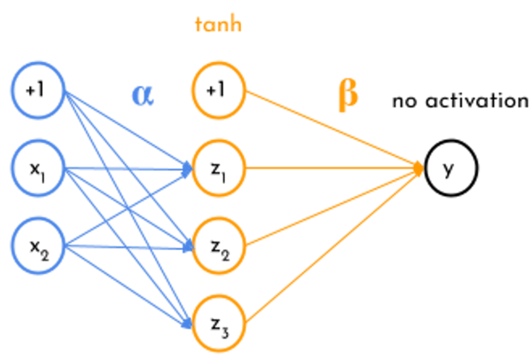

Figure 1: A One Hidden Layer Neural Network

For example, using the network shown in Figure 1, we can calculate the value of z_1 in two steps. First, we need to multiply the input values by their weights and sum them together. Assume for the sake of notation that the bias term (marked as +1 in figure, is x_0)

$$a_1 = \sum_{i=0}^{2} \alpha_{1,i} x_i \tag{1}$$

$$z_1 = \tanh a_1 \tag{2}$$

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

   Similar to how an intercept term in linear regression allows it to better fit data, the bias term helps the neural network better fit its data as well.

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

   A neural network with only linear activation functions would be no different than a linear regression. (Try forward propagating with only linear functions on the given example)

2 Backward Propagation Explained

Backward propagation Given a Neural Network and a corresponding loss function $J(\theta)$, back-propagation 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 $\frac{\partial J}{\partial \alpha_{ij}}$ and $\frac{\partial J}{\partial \beta_{kj}}$, 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.

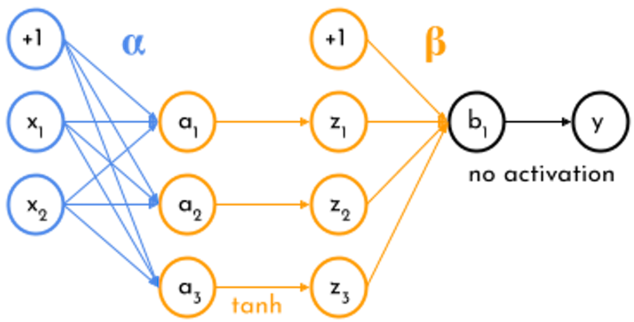

Figure 2: Extended Version of Previous Neural Network

Although back propagation may seem really tricky, it becomes very simple if you break up the process into individual layers. Above in Figure 2 is an extended version of the previous neural network above. Here, each layer is broken into its linear combination stage and its activation stage. Here is an example of breaking down the partial loss with respect to the weight $\alpha_{1,1}$:

$$\frac{\partial J}{\partial \alpha_{1,1}} = \frac{\partial J}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial \alpha_{1,1}} \tag{3}$$

$$= \frac{\partial J}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial b_1} \frac{\partial b_1}{\partial \alpha_{1,1}} \tag{4}$$

$$= \frac{\partial J}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial b_1} \frac{\partial b_1}{\partial z_1} \frac{\partial z_1}{\partial \alpha_{1,1}} \tag{5}$$

$$= \frac{\partial J}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial b_1} \frac{\partial b_1}{\partial z_1} \frac{\partial z_1}{\partial a_1} \frac{\partial a_1}{\partial \alpha_{1,1}} \tag{6}$$

1. Many gradients are calculated in back propagation. Which of these gradients directly update the weights? Do not include intermediate value(s) used to calculate these gradient(s).
   The gradients with respect to α and β are used in updating. The rest are intermediate values used to calculate these two gradients

3 Neural Network Example Explained

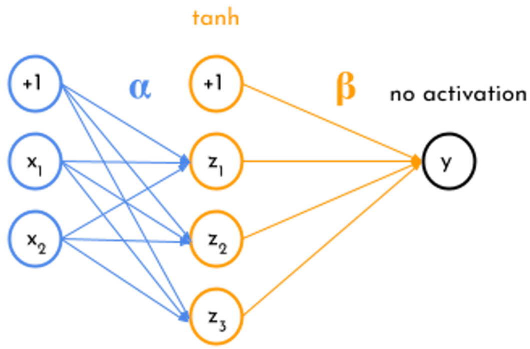

Figure 3: Neural Network For Example Questions

Network Overview Consider the neural network with one hidden layer shown in Figure 3. 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:

$$\mathbf{x} = [x_0, x_1, x_2]^T \tag{7}$$

Linear combination at the first (hidden) layer:

$$a_j = \sum_{i=0}^{2} \alpha_{j,i} \cdot x_i, \;\; \forall j \in \{1, \ldots, 3\} \tag{8}$$

Activation at the first (hidden) layer:

$$z_j = \tanh(a_j) = \frac{e^{a_j} - e^{-a_j}}{e^{a_j} + e^{-a_j}}, \;\; \forall j \in \{1, \ldots, 3\} \tag{9}$$

Linear combination at the second (output) layer:

$$\hat{y} = \sum_{j=0}^{3} \beta_j \cdot z_j, \tag{10}$$

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, $\ell(\hat{y}, y)$:

$$\ell(\hat{y}, y) = \frac{1}{2}(\hat{y} - y)^2 \tag{11}$$

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. **Scalar Form:** Given $x_1 = 1$, $x_2 = 2$,

   - Forward: What are the values of a_1, ℓ?

$$a_1 = \sum_{i=0}^{2} \alpha_{1,i} x_i = \qquad z_1 = \qquad \hat{y} =$$

$$a_2 = \sum_{i=0}^{2} \alpha_{2,i} x_i = \qquad z_2 = \qquad \ell =$$

$$a_3 = \sum_{i=0}^{2} \alpha_{3,i} x_i = \qquad z_3 =$$

$$a_1 = \sum_{i=0}^{2} \alpha_{1,i} x_i = 5 \qquad z_1 = 0.99991 \qquad \hat{y} = 4.91807$$

$$a_2 = \sum_{i=0}^{2} \alpha_{2,i} x_i = 3 \qquad z_2 = 0.99505 \qquad \ell = 1.83950$$

$$a_3 = \sum_{i=0}^{2} \alpha_{3,i} x_i = 2 \qquad z_3 = 0.96403$$

- Backward: What are the values of $\frac{\partial \ell}{\partial \alpha_{1,1}}$, $\frac{\partial \ell}{\partial \beta_1}$ **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

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

$$\begin{aligned}
\frac{\partial \ell}{\partial \alpha_{1,1}} &= \frac{\partial \ell}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial z_1} \frac{\partial z_1}{\partial a_1} \frac{\partial a_1}{\partial \alpha_{1,1}} \\
&= (\hat{y} - y)\beta_1(1 - \tanh(a_1)^2)x_1 \\
&= (4.91807 - 3) * 1 * (1 - \tanh(5)^2) * 1 \\
&= 0.000348
\end{aligned}$$

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

As a reminder, we were given that $\ell = \frac{1}{2}(\hat{y} - y)^2$ and $\hat{y} = \sum_{j=0}^{3} \beta_j z_j$

So we can calculate:

$$\begin{aligned}
\frac{\partial \ell}{\partial \beta_1} &= \frac{\partial \ell}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial \beta_1} \\
&= \frac{\partial}{\partial \hat{y}} \left[\frac{1}{2}(\hat{y} - y)^2 \right] \frac{\partial}{\partial \beta_1} \left[\sum_{j=0}^{3} \beta_j z_j \right] \\
&= (\hat{y} - y)\, z_1 = (4.91807 - 3) * 0.99991 = 1.9179
\end{aligned}$$

2. **Vector Form:** The vector form of forward computation is:

$$\begin{aligned}\mathbf{a} &= \hat{\mathbf{x}}\boldsymbol{\alpha}^\top \\ \mathbf{z} &= \tanh(\mathbf{a}) \\ \hat{y} &= \hat{\mathbf{z}}\boldsymbol{\beta}^\top\end{aligned} \tag{12}$$

Given $\mathbf{x} = \begin{bmatrix}0 & 1\end{bmatrix}$,

- Forward: Find ℓ?

  Denote $\hat{\mathbf{x}}$ as the augmented $\mathbf{x}$ by appending 1 to the front to obtain a compact representation that includes the bias term.

  $\mathbf{a} = \hat{\mathbf{x}}\boldsymbol{\alpha}^\top = \begin{bmatrix}1 & 0 & 1\end{bmatrix}\begin{bmatrix}0 & 2 & 0\\1 & 1 & 2\\2 & 0 & 0\end{bmatrix} = \begin{bmatrix}2 & 2 & 0\end{bmatrix}$

  $\mathbf{z} = \tanh \mathbf{a} = \begin{bmatrix}0.96403 & 0.96403 & 0\end{bmatrix}$

  We apply the same trick to $\mathbf{z}$ to obtain $\hat{\mathbf{z}} = \begin{bmatrix}1 & 0.96403 & 0.96403 & 0\end{bmatrix}$

  $\hat{y} = \hat{\mathbf{z}}\boldsymbol{\beta}^\top = \begin{bmatrix}1 & 0.96403 & 0.96403 & 0\end{bmatrix}\begin{bmatrix}0\\1\\2\\2\end{bmatrix} = 3 \times 0.96403 = 2.89209$

  $\ell = \frac{1}{2}(\hat{y} - 3)^2 = 0.005822284$

- Backward: What are the values of $\frac{\partial \ell}{\partial \boldsymbol{\alpha}}$, $\frac{\partial \ell}{\partial \boldsymbol{\beta}}$?

  Denote $\hat{\boldsymbol{\beta}}$ as $\boldsymbol{\beta}$ without the first entry.

$$\begin{aligned}
\frac{\partial \ell}{\partial \mathbf{a}} &= \frac{\partial \ell}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial \mathbf{z}} \frac{\partial \mathbf{z}}{\partial \mathbf{a}} \\
&= (\hat{y} - y) \cdot \hat{\boldsymbol{\beta}} \cdot \operatorname{diag}(1 - \tanh^2(\mathbf{a})) \\
&= (\hat{y} - y) \cdot \hat{\boldsymbol{\beta}} \odot (1 - \mathbf{z}^2) \qquad\qquad (\odot \text{ is element-wise multiplication}) \\
&= -0.10791 \begin{bmatrix} 1 & 2 & 2 \end{bmatrix} \begin{bmatrix} 1 - \tanh^2(a_1) & 0 & 0 \\ 0 & 1 - \tanh^2(a_2) & 0 \\ 0 & 0 & 1 - \tanh^2(a_3) \end{bmatrix} \\
&= -0.10791 \begin{bmatrix} 1 & 2 & 2 \end{bmatrix} \begin{bmatrix} 0.07065082 & 0 & 0 \\ 0 & 0.07065082 & 0 \\ 0 & 0 & 1 \end{bmatrix}
\end{aligned}$$

$$\begin{aligned}
\frac{\partial \ell}{\partial \alpha_{ji}} &= \frac{\partial \ell}{\partial a_j} \frac{\partial a_j}{\partial \alpha_{ji}} \\
&= \frac{\partial \ell}{\partial a_j} x_i
\end{aligned}$$

$$\begin{aligned}
\frac{\partial \ell}{\partial \boldsymbol{\alpha}} &= \frac{\partial \ell}{\partial \mathbf{a}} \hat{\mathbf{x}}^T \\
&= \begin{bmatrix} -0.00762444 & 0 & -0.00762444 \\ -0.01524889 & 0 & -0.01524889 \\ -0.21583452 & 0 & -0.21583452 \end{bmatrix}
\end{aligned}$$

$$\frac{\partial \ell}{\partial \boldsymbol{\beta}} = \frac{\partial \ell}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial \boldsymbol{\beta}} = (\hat{y} - 3)\hat{\mathbf{z}}^T = \begin{bmatrix} -0.10791 & -0.10402 & -0.10402 & 0 \end{bmatrix}$$

Given $\mathbf{x} = \begin{bmatrix} 2 \\ 3 \end{bmatrix}$,

- Forward: Find ℓ?

$$\mathbf{a} = \boldsymbol{\alpha}\hat{\mathbf{x}} = \begin{bmatrix} 0 & 1 & 2 \\ 2 & 1 & 0 \\ 0 & 2 & 0 \end{bmatrix} \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix} = \begin{bmatrix} 8 \\ 4 \\ 4 \end{bmatrix}$$

$$\mathbf{z} = \tanh \mathbf{a} = \begin{bmatrix} 0.9999 \\ 0.9993 \\ 0.9993 \end{bmatrix}$$

$$\hat{\mathbf{z}} = \begin{bmatrix} 1 \\ 0.9999 \\ 0.9993 \\ 0.9993 \end{bmatrix}$$

$$\hat{y} = \boldsymbol{\beta}\hat{\mathbf{z}} = \begin{bmatrix} 0 & 1 & 2 & 2 \end{bmatrix} \begin{bmatrix} 1 \\ 0.9999 \\ 0.9993 \\ 0.9993 \end{bmatrix} = 4.997316$$

$$\ell = \tfrac{1}{2}(\hat{y} - 3)^2 = 1.9946$$

- Backward: What are the values of $\frac{\partial \ell}{\partial \boldsymbol{\alpha}}$, $\frac{\partial \ell}{\partial \boldsymbol{\beta}}$?

Define $\hat{\boldsymbol{\beta}}$ similarly.

$$\frac{\partial \ell}{\partial \mathbf{a}} = \frac{\partial \ell}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial \mathbf{z}} \frac{\partial \mathbf{z}}{\partial \mathbf{a}} = (\hat{y} - y) \cdot \hat{\boldsymbol{\beta}}^T \odot (1 - \mathbf{z}^2) = \begin{bmatrix} 8.992 * 10^{-7} \\ 5.357 * 10^{-3} \\ 5.357 * 10^{-3} \end{bmatrix}$$

$$\frac{\partial \ell}{\partial \alpha_{ji}} = \frac{\partial \ell}{\partial a_j} \frac{\partial a_j}{\partial \alpha_{ji}} = \frac{\partial \ell}{\partial a_j} x_i$$

$$\frac{\partial \ell}{\partial \alpha} = \frac{\partial \ell}{\partial \mathbf{a}} \hat{\mathbf{x}}^T = \begin{bmatrix} 8.992 * 10^{-7} \\ 5.357 * 10^{-3} \\ 5.357 * 10^{-3} \end{bmatrix} \begin{bmatrix} 1 & 2 & 3 \end{bmatrix} = \begin{bmatrix} 8.998 * 10^{-7} & 1.798 * 10^{-6} & 2.698 * 10^{-6} \\ 5.357 * 10^{-3} & 1.071 * 10^{-2} & 1.607 * 10^{-2} \\ 5.357 * 10^{-3} & 1.071 * 10^{-2} & 1.607 * 10^{-2} \end{bmatrix}$$

$$\frac{\partial \ell}{\partial \boldsymbol{\beta}} = \frac{\partial \ell}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial \boldsymbol{\beta}} = (\hat{y} - y)\hat{\mathbf{z}}^T = \begin{bmatrix} 1.9973 & 1.9973 & 1.9960 & 1.9960 \end{bmatrix}$$

4 CNNs Explained

4.1 Overview

4.1.1 Motivation

When working with large inputs, such as images, we often do not have enough data to train a fully-connected dense network. Using filters slide over the input via convolution allows us to use fewer parameters while still processing the entire input through multiple layers, allowing us to train networks with much less data.

When working with certain types of data, we are unconcerned with where in an input a particular feature appears. If we only care about whether a feature is present or not, we can take advantage of the *translation invariance* provided by using convolutional layers. This allows us to use the same filter for detecting a feature regardless of where in the image the feature occurs.

4.1.2 Input

While CNNs can be applied to many different types of data, we often consider them in terms of images. Simple images are matrices (or *2-tensors*), with the dimensions *height* $\times$ *width*. An example would be a grayscale image. More complex images are represented by *3-tensors*, with dimensions $height \times width \times channels$. For example, an RGB image may have 3 channels: one for each of red, green, and blue pixel values. Each of these channels is a 2-dimensional matrix with dimensions *height* $\times$ *width* containing a value for each pixel in the image.

4.1.3 Kernels and Filters

Similar to inputs, kernel can take on many different forms. We'll usually consider kernels represented by matrices, with shape *height* $\times$ *width*. We often work with square kernels. We also generally work with kernels with odd dimensions: e.g. 3×3 or 5×5 kernels.

You may have heard the terms filter and kernel used interchangeably. The term kernel refers to a 2-tensor, or matrix, of weights. The term filter refers to the set of kernels being used. If only one is used, as you'll usually see in this class, then the terms filter and kernel are interchangeable.

4.1.4 Stride

The stride refers to how we slide our kernel across a given input. If at each step, we move our kernel by one position on the input, then our stride is 1. If we skip a possible position and move our kernel by 2 spots each time, our stride is 2.

4.1.5 Padding

When convolving a kernel with an image, we often want to make sure each part of the kernel is applied to each part of the image. This presents a problem at the borders of the image: for example, only the top part of the kernel will ever interact with the top part of the image. We resolve this issue by *padding* the input: adding fake values (according to one of a variety of rules) around the borders of the image to make it artificially larger, such that every part of our kernel can now interact with every part of the real image data. Padding is usually applied equally to all sides of the image: if we pad an image of size 10×10 with a padding of 2, we have an augmented image of size 14×14.

4.1.6 Output Size

Input size, filter size, stride, and padding all come together when determining the output size of a given layer. Below, we build up the formula for output size by adding these one-by-one. The formula we'll construct involves a single square kernel for the filter and uniform padding and stride values with a 3-channel image.

- Suppose we have an image of size $H \times W$ and a kernel of size 1×1. Our output is naturally of size $H \times W$.
- Suppose we have an image of size $H \times W$ and a kernel of size $K \times K$. Our output is now of size $(H - K + 1) \times (W - K + 1)$.
- Suppose we have an image of size $H \times W$, a kernel of size $K \times K$, and a padding P. Our output is now of size $(H + 2P - K + 1) \times (W + 2P - K + 1)$.
- Suppose we have an image of size $H \times W$, a kernel of size $K \times K$, a padding P, and a stride S. Our output is now of size $(\lfloor \frac{H+2P-K}{S} \rfloor + 1) \times (\lfloor \frac{W+2P-K}{S} \rfloor + 1)$.
- Suppose we have an image of size $H \times W \times D$, a kernel of size $K \times K$, a padding P, and a stride S. Our output is now of size $(\lfloor \frac{H+2P-K}{S} \rfloor + 1) \times (\lfloor \frac{W+2P-K}{S} \rfloor + 1) \times D$.

We can extend this formula to work with multi-kernel filters, non-square kernels, and other parameter options.

2	3	-1	2
4	1	0	1
-2	1	3	2
-1	-2	0	5

(a) X

1	0	1
-1	0	-1
1	0	1

(b) F

4.2 Problems

(a) Let X be convolved with F using no padding and a stride of 1 to produce an output Y. What is value of the output Y?

$\mathbf{Y} = \begin{bmatrix} -2 & 6 \\ 2 & 2 \end{bmatrix}$. Each of the values correspond to applying the filter to the top left, top right, bottom left, and bottom right parts of the image.

Top left: $-2 = (1*2+0*3+1*-1)+(-1*4+0*1+-1*0)+(1*-2+0*1+1*3)$.

Top right: $6 = (1*3+0*-1+1*2)+(-1*1+0*0+-1*1)+(1*1+0*3+1*2)$

Bottom left: $2 = (1*4+0*1+1*0)+(-1*-2+0*1+-1*3)+(1*-1+0*-2+1*0)$

Bottom right: $2 = (1*1+0*0+1*1)+(-1*1+0*3+-1*2)+(1*-2+0*0+1*5)$

(b) Suppose you had an input feature map of size (height $\times$ width) 9×6 and filter size 5×5, using a padding of 4 and a stride of 3, what would be the resulting output size? Write your answer in the format height $\times$ width.

5×4.

$$
\begin{aligned}
H \times W &= 9 \times 6 \\
H + 2P \times W + 2P &= 17 \times 14 \\
H + 2P - K \times W + 2P - K &= 12 \times 9 \\
\left\lfloor \frac{H + 2P - K}{S} \right\rfloor \times \left\lfloor \frac{W + 2P - K}{S} \right\rfloor &= 4 \times 3 \\
\left\lfloor \frac{H + 2P - K}{S} \right\rfloor + 1 \times \left\lfloor \frac{W + 2P - K}{S} \right\rfloor + 1 &= 5 \times 4
\end{aligned}
$$