Backpropagation

Zheguang Zhao

We use a for activations, x for the input examples, y for the output, $\mathbf{w}$ for weights, b for bias, L for loss and J for cost. We use round brackets for indices of input examples, square brackets for indices of network layers, and triangular brackets for time steps in a RNN layer. For example, the j-th activation at the l-th layer induced by the m-th input example is $a_j^{[l](m)}$.

The activation function is

$$\begin{aligned} z_j^{[l](m)} &= w_j^{[l]T} x^{(m)} + b_j \\ a_j^{[l](m)} &= \sigma(z_j^{[l](m)}) \end{aligned}$$

We often call z the linear part.

1 Logistic regression

Let's first consider binary classification using logistic regression model. This can be seen as a single-layer, single-activation neural network. So the activation function is just

$$\begin{aligned} z &= \mathbf{w} \cdot \mathbf{x} + b \\ a &= \sigma(z) \end{aligned}$$

The activation is interpreted as the conditional probability that the class is 1:

$$P(y = 1|x) := a$$

The output function can then be defined based on this conditional probability a:

$$\hat{y} = \begin{cases} 1 & a > 1/2 \\ 0 & \text{else} \end{cases}$$

The output is correct when $\hat{y} = y$. But how correct is correct? When $y = 1$, we would feel the activation of $a = 0.99$ better than $a = 0.51$, because the

former is more "certain" about the correct prediction. Similarly, the output is wrong when $\hat{y} \neq y$. But how wrong is wrong? When $y = 1$, we would feel that the activation of $a = 0.4$ better than the $a = 0.1$, because the latter is regretfully more "certain" about the wrong prediction.

It is this notion of "certainty about the correct prediction" that we want to quantify as the (opposite of) loss. Higher the certainty about the correct prediction, lower the loss.

$$\begin{aligned} -L(y, \hat{y}) &:= y \log(a) + (1 - y) \log(1 - a) \\ L(y, \hat{y}) &= -\big(y \log(a) + (1 - y) \log(1 - a)\big) \end{aligned}$$

Note that the loss is by definition a measure of "distance" between y and $\hat{y}$, but the function uses a and y.

The loss is calculated based on just one input example. The total cost of the model is just the "average loss" from all examples:

$$J := \frac{1}{m} \sum_{i}^{m} L(y^{(i)}, \hat{y}^{(i)}) = \frac{\mathbf{L} \cdot \mathbf{1}}{m}$$

1.1 Computation graph

Now we have just defined a **computation graph for cost** which starts from w and b, and then with input x, to a, and then to z, then to a, then to $\hat{y}$, then to L, and finally to J. Note this computation graph is just a function with one set of parameters w and b, but is invoked m times because there are m inputs x and labels y, and so results in m number of z, a, $\hat{y}$ and L, and finally congregating to just one J.

There is also a **computation graph for prediction**, which is the same but without labels y and so no L and J, but only with output $\hat{y}$.

For training, we consider making m predictions at one "batch" so we can get an average sense of the errors. But for prediction, we usually consider making one prediction at a time. So the computation graph for cost can be thought of as the computation graph for prediction being copied m times, while keeping the same parameters $\mathbf{w}$ and b.

1.2 Reducing mis-prediction

With randomly initialized activation function parameters, we will have some mis-predictions. So J will have some corresponding high value. Reducing

mis-predictions corresponds to reducing cost.[1] To reduce the cost, we need to adjust the individual loss L, which means to adjust the prediction $\hat{y}$, and that means the activation a, and that means the activation function parameters w and b. This backward chain of adjustment can be seen as going backward from the computation graph for training starting from J and finishing at w and b.

The most efficient adjustment we seek is along the direction of the gradient of the cost function. That means we need to find the derivatives of $\frac{\partial J}{\partial w}$ and $\frac{\partial J}{\partial b}$. What's the dimension of the gradient? As many dimensions as there are of w.

To find the gradient we need to apply the chain rule. We first note that J is a function of $L^{(i)}$s:

$$\mathrm{d}\mathbf{L} := \frac{\mathrm{d}J}{\mathrm{d}\mathbf{L}} = \left(\frac{\partial J}{\partial L^{(1)}}, \cdots, \frac{\partial J}{\partial L^{(m)}}\right) = \frac{\mathbf{1}}{m}$$

where

$$\frac{\partial J}{\partial L^{(i)}} = \frac{1}{m}, \text{ for } i = 1, \cdots, m$$

Then L is a function of a, so:

$$\frac{\mathrm{d}L}{\mathrm{d}a} = -\left(\frac{y}{a} - \frac{1-y}{1-a}\right)$$

a is a function of z, so:

$$\frac{\mathrm{d}a}{\mathrm{d}z} = a(1-a)$$

z is a linear function of w and b, so:

$$\frac{\partial z}{\partial \mathbf{w}} = \mathbf{x}$$
$$\frac{\partial z}{\partial b} = 1$$

Note that the weights $\mathbf{w}$ are vectors whose dimension equals that of the input $\mathbf{x}$, such as the pixel dimension of an input image.

So chain rule gives us:

$$\mathrm{d}\mathbf{z} := \frac{\mathrm{d}J}{\mathrm{d}\mathbf{z}} = \frac{\mathrm{d}\mathbf{a}}{\mathrm{d}\mathbf{z}}\frac{\mathrm{d}\mathbf{L}}{\mathrm{d}\mathbf{a}}\frac{\mathrm{d}J}{\mathrm{d}\mathbf{L}}$$
$$\frac{\mathrm{d}J}{\mathrm{d}(\mathbf{w}, b)} = \left(\frac{\partial \mathbf{z}}{\partial \mathbf{w}} \cdot \mathrm{d}\mathbf{z}\,, \frac{\partial \mathbf{z}}{\partial b} \cdot \mathrm{d}\mathbf{z}\right)$$

[1] This error reduction is true at least for the training set, and we can only hope it generalizes to unseen examples.

It's worth unpacking the notations here. It says the gradient of J w.r.t. $\mathbf{w}$ is the dot product $\frac{\partial \mathbf{z}}{\partial \mathbf{w}} \cdot \mathrm{d}\mathbf{z}$. We should expect it to have the dimension of $\mathbf{w}$. What does the dot product mean? It means for all the m parts in gradient $\mathrm{d}\mathbf{z}$, sum up all the effects caused by a tiny change of $\mathbf{w}$. But where do the m parts of gradient $\mathrm{d}\mathbf{z}$ come from? They come from the m parts of gradient $\mathrm{d}\mathbf{L}$. Then why do we sum? Well, because the same weights are used for m different input examples to make predictions:

$$
\begin{aligned}
\mathrm{d}z^{(i)} &:= \frac{\mathrm{d}a^{(i)}}{\mathrm{d}z^{(i)}} \frac{\mathrm{d}L^{(i)}}{\mathrm{d}a^{(i)}} \frac{\partial J}{\partial L^{(i)}} \\
\frac{\partial J}{\partial \mathbf{w}} &= \frac{\partial z^{(1)}}{\partial \mathbf{w}} \mathrm{d}z^{(1)} + \cdots + \frac{\partial z^{(m)}}{\partial \mathbf{w}} \mathrm{d}z^{(m)} \\
&= \begin{bmatrix} \frac{\partial z^{(1)}}{\partial \mathbf{w}} & \cdots & \frac{\partial z^{(m)}}{\partial \mathbf{w}} \end{bmatrix} \cdot \begin{bmatrix} \mathrm{d}z^{(1)} \\ \vdots \\ \mathrm{d}z^{(m)} \end{bmatrix} \\
&= \frac{\partial \mathbf{z}}{\partial \mathbf{w}} \cdot \mathrm{d}\mathbf{z}
\end{aligned}
$$

Here, we can visualize $\frac{\partial J}{\partial \mathbf{w}}$ as the product of a matrix of column vectors $\left\{ \frac{\partial z^{(i)}}{\partial \mathbf{w}} \right\}$ and a column vector $\mathrm{d}\mathbf{z}$.