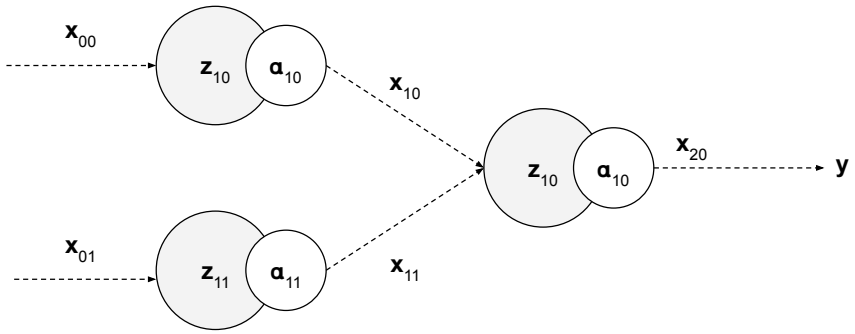
Backpropagation Derivation

Pradeep Ranganathan

October 20, 2021

In this document, we derive back-prop equations for a simple 2-hidden-layer neural-net and show how we can take advantage of repeated expressions in partial derivative evaluation to efficiently compute the gradient via back-propagation.

1 The Model



Each neural unit computes a linear function of its input $z = wx + b$ and then sends the linear output through a non-linear activation function $\alpha(\cdot)$. We assume that all neural units use the same activation function, α, as is common in practice. Hence for the first layer that operates on the inputs, x_{10} and x_{11}, we have:

$$
\begin{aligned}
z_{10} &= w_{10}x_{00} + b_{10} \\
x_{10} &= \alpha(z_{10}) \\
z_{11} &= w_{11}x_{01} + b_{11} \\
x_{11} &= \alpha(z_{11})
\end{aligned}
$$

For the second layer, the neural unit operates on a vector of inputs, $\mathbf{x}_1 = [x_{10}\ x_{11}]^\top$. For computation of the back-prop equations we expand the vector linear operation into it constituent scalar parts:

$$
\begin{aligned}
z_{20} &= \mathbf{w}_{20}^\top \mathbf{x}_1 + \mathbf{b}_{20} \\
z_{20} &= (w_{20}^{[0]}x_{10} + b_{20}^{[0]}) + (w_{20}^{[1]}x_{11} + b_{20}^{[1]}) \\
x_{20} &= \alpha(z_{20})
\end{aligned}
$$

Finally, the output, y, is the output of the last neuron:

$$
y = x_{20}
$$

2 Back-prop equations

2.1 Partial derivative w.r.t w_{10}

Using ∂ to denote the operator $\frac{\partial}{\partial w_{10}}$:

$$
\begin{aligned}
\frac{\partial}{\partial w_{10}} y &= \partial x_{20} \\
&= \partial \alpha(z_{20}) \\
&= \alpha'(z_{20}) \cdot \partial z_{20} \\
&= \alpha'(z_{20}) \cdot \partial \left\{ (w_{20}^{[0]} x_{10} + b_{20}^{[0]}) + (w_{20}^{[1]} x_{10} + b_{20}^{[1]}) \right\} \\
&= \alpha'(z_{20}) \cdot \left\{ w_{20}^{[0]} \partial x_{10} + w_{20}^{[1]} \partial x_{11} \right\} \\
&= \alpha'(z_{20}) \cdot \left\{ w_{20}^{[0]} \partial \alpha(z_{10}) + w_{20}^{[1]} \partial \alpha(z_{11}) \right\} \\
&= \alpha'(z_{20}) \cdot \left\{ w_{20}^{[0]} \alpha'(z_{10}) \partial (w_{10} x_{00} + b_{10}) + w_{20}^{[1]} \alpha'(z_{11}) \partial (w_{11} x_{01} + b_{11}) \right\} \\
&= \alpha'(z_{20}) \cdot \left\{ w_{20}^{[0]} \alpha'(z_{10}) x_{00} \partial w_{10} + 0 \right\} \\
&= \alpha'(z_{20}) \cdot w_{20}^{[0]} \alpha'(z_{10}) x_{00}
\end{aligned}
$$

2.2 Partial derivative w.r.t w_{11}

Using ∂ to denote the operator $\frac{\partial}{\partial w_{11}}$:

$$
\begin{aligned}
\frac{\partial}{\partial w_{11}} y &= \partial x_{20} \\
&= \partial \alpha(z_{20}) \\
&= \alpha'(z_{20}) \cdot \partial z_{20} \\
&= \alpha'(z_{20}) \cdot \partial \left\{ (w_{20}^{[0]} x_{10} + b_{20}^{[0]}) + (w_{20}^{[1]} x_{10} + b_{20}^{[1]}) \right\} \\
&= \alpha'(z_{20}) \cdot \left\{ w_{20}^{[0]} \partial x_{10} + w_{20}^{[1]} \partial x_{11} \right\} \\
&= \alpha'(z_{20}) \cdot \left\{ w_{20}^{[0]} \partial \alpha(z_{10}) + w_{20}^{[1]} \partial \alpha(z_{11}) \right\} \\
&= \alpha'(z_{20}) \cdot \left\{ w_{20}^{[0]} \alpha'(z_{10}) \partial (w_{10} x_{00} + b_{10}) + w_{20}^{[1]} \alpha'(z_{11}) \partial (w_{11} x_{01} + b_{11}) \right\} \\
&= \alpha'(z_{20}) \cdot \left\{ 0 + w_{20}^{[1]} \alpha'(z_{11}) \partial (w_{11} x_{01} + b_{11}) \right\} \\
&= \alpha'(z_{20}) \cdot w_{20}^{[1]} \alpha'(z_{11}) x_{01}
\end{aligned}
$$

2.3 Other partial derivatives

The other partial derivatives can be computed in a similar manner. We summarize all partial derivatives w.r.t the parameters of the neural-net below:

$$
\begin{aligned}
\frac{\partial}{\partial b_{20}^{[0]}} y &= \alpha'(z_{20}) \\
\frac{\partial}{\partial b_{20}^{[1]}} y &= \alpha'(z_{20}) \\
\frac{\partial}{\partial w_{20}^{[0]}} y &= \alpha'(z_{20}) \cdot x_{10} \\
\frac{\partial}{\partial w_{20}^{[1]}} y &= \alpha'(z_{20}) \cdot x_{11} \\
\frac{\partial}{\partial b_{10}} y &= \alpha'(z_{20}) \cdot w_{20}^{[0]} \alpha'(z_{10}) \\
\frac{\partial}{\partial w_{10}} y &= \alpha'(z_{20}) \cdot w_{20}^{[1]} \alpha'(z_{10}) x_{00} && \text{(derived in 2.1)} \\
\frac{\partial}{\partial b_{11}} y &= \alpha'(z_{20}) \cdot w_{20}^{[1]} \alpha'(z_{11}) \\
\frac{\partial}{\partial w_{11}} y &= \alpha'(z_{20}) \cdot w_{20}^{[1]} \alpha'(z_{11}) x_{01} && \text{(derived in 2.2)}
\end{aligned}
$$

3 Efficient back-prop

After examining the above expressions it should be apparent that the sub-expressions $\alpha'(z_{20})$, $\alpha'(z_{20}) \cdot w_{20}^{[0]}$ and $\alpha'(z_{20}) \cdot w_{20}^{[1]}$ are shared amongst different sub-sets of partial derivative expressions.

By collecting common sub-expressions from the partial derivative expressions and creating an directed acyclic graph using *is-a-subexpression-of* relations, it is possible to produce a series of compounding computations that evaluate each partial derivative with a constant amount of computation on top of existing expressions. This lead-us to a back-to-front evaluation strategy where sub-expressions are computed and reused, starting from the last-layer to the first and hence the term *back-propagation of errors*.