Back Propagation - Stanford CS231N

Name: Eli Andrew

- **Problem Statement:** given some function $f(x)$ compute the gradient of f at x ($\nabla f(x)$)
- Derivative of a function f with respect to some variable x is how **sensitive** the whole expression is to that value
- Derivatives tell you nothing about large changes in the inputs of a function: the are only informative for tiny, infinitesimally small changes on the inputs
- **Backpropagation Intuition:**
  - The entire circuit or function "wants" to output a higher value
  - Therefore each gate or node "wants" its inputs to change according to its gradient (-4 gradient means gate wants its inputs to decrease, because of negative sign, and with a force of 4)
  - Backprop can be thought of as the different gates communicating with one another, through gradient signal, whether they want their outputs to increase or decrease (and how strongly) to make the final output value higher
- **Technical Implementation of Backprop:**
  - It is always helpful to breakdown the forward pass into stages that are easily backpropped through
  - The details of how backprop is performed, and how we break up the stages of the forward pass (what we view as the gates / nodes), is a matter of convenience
  - It helps to be aware of which parts of the expression have easy local gradients, so that they can be chained together with the least amount of code and effort
- **Patterns in backward flow:**
  - **Add Gate**
    - The add gate always distributes its gradient evenly among all inputs
    - This follows from the derivatives (ex for 2 inputs a and b)

$$f(a,b) = a + b$$
$$\frac{df}{da} = 1$$
$$\frac{df}{db} = 1$$

* So the gradient flowing back into the add gate (the gradient from the part of the system after this) flows evenly to each input

$$a)\frac{df}{da} * \frac{dy}{df} = \frac{dy}{df}$$
$$b)\frac{df}{db} * \frac{dy}{df} = \frac{dy}{df}$$

- **Max Gate**
  * The max gate always distributes its whole gradient to the maximum input
  * Consider again the derivatives (ex for 2 inputs a and b)

$$f(a,b) = max(a,b) if a > b = a, \frac{df}{da} = 1, \frac{df}{db} = 0 if b > a = b, \frac{df}{da} = 0, \frac{df}{db} = 1$$

  * So, the gradient updates for each input are:

$$a)\frac{df}{da} * \frac{dy}{df} = 0 if a < b, \frac{dy}{df} if a > b b)\frac{df}{db} * \frac{dy}{df} = 0 if b < a, \frac{dy}{df} if b > a$$

- **Multiply Gate**
  * The multiply gate always distributes its gradient in a less intuitive way
  * Again, consider the derivatives (ex this time for 3 inputs a, b, and c)

$$f(a,b,c) = a * b * c$$
$$\frac{df}{da} = b * c$$
$$\frac{df}{db} = a * c$$
$$\frac{df}{dc} = a * b$$

  * So, the gradient updates for each input are:

$$a)\frac{df}{da} * \frac{dy}{df} = b * c * \frac{dy}{df}$$
$$b)\frac{df}{db} * \frac{dy}{df} = a * c * \frac{dy}{df}$$
$$c)\frac{df}{dc} * \frac{dy}{df} = a * b * \frac{dy}{df}$$

  * So, the multiply gate scales the gradient received by each input by the value of all the other inputs combined
- **Un-intuitive effects of backprop**: Notice that a multiply gate will assign a relatively large gradient to the small input, and a relatively small gradient to the largest input. As a result the actual scaling of your data has a large effect on gradients, which is one reason why pre-processing is so important!

- **Tensor Derivatives with Backprop**
  - If $f : R^{N_1 \times \cdots \times N_{D_x}} \rightarrow R^{M_1 \times \cdots \times M_{D_y}}$
  - Input to f is D_x dimensional tensor of shape $N_1 \times \cdots \times N_{D_x}$
  - Output of f is D_y dimensional tensor of shape $M_1 \times \cdots \times M_{D_y}$
  - If $y = f(x)$ then $\frac{\partial y}{\partial x}$ is shape: $(M_1 \times \cdots \times M_{D_y}) \times (N_1 \times \cdots \times N_{D_x})$
  - Generalized Jacobian y can be thought of as the generalization of a matrix, where each row has the same shape as y, and each column has the same shape as x
  - Let $i \in Z^{D_y}$ and $j \in Z^{D_x}$
  - Then: $(\frac{\partial y}{\partial x})_{i,j} = \frac{\partial y_i}{\partial x_j}$
  - Where y_i and x_i are scalars, so $\frac{\partial y_i}{\partial x_j}$ is also a scalar
  - Just like standard Jacobian, the generalized Jacobian tells relative rate of change between all elements of x and all elements of y
  - Relationship here is: $x \rightarrow x + \Delta x \Rightarrow y + \frac{\partial y}{\partial x} \Delta x$
  - Where $\frac{\partial y}{\partial x} \Delta x$ is a generalized matrix-vector multiply which gives tensor of shape $M_1 \times \cdots \times M_{D_y}$