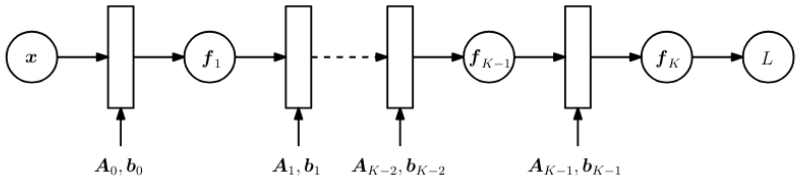
Backpropogation

Jack David Carson - February 1, 2025

Finding good parameters for a machine learning model relies on the fact we can compute the gradient of a learning objective with respect to the parameters of a model. The textbf{backpropogation} algorithm is an efficient way to compute the gradient of an error function with respect to the model parameters.

In deep learning, the chain rule is used to the extreme where the function $\boldsymbol{y}$ is computed as a many level function composition $\boldsymbol{y} = (f_K \circ f_{K-1} \circ \cdots \circ f_1)(\boldsymbol{x})$ for x inputs and y observations.



In a neural network with multiple layers we have functions $f_i(\boldsymbol{x}_{i-1}) = \sigma(A_{i-1}\boldsymbol{x}_{i-1} + \boldsymbol{b}_{i-1})$ in layer i for **sigmoid activation function** such as $\sigma(\boldsymbol{x}) = \frac{1}{1+e^{-x}}$. For instance, we can have loss function

$$\mathcal{L}(\boldsymbol{\theta}) = \left\| \boldsymbol{y} - \boldsymbol{f}_K(\boldsymbol{\theta}, \boldsymbol{x})^2 \right\|$$

where $\boldsymbol{\theta} = \{A_0, \boldsymbol{b}_0, ..., A_{K-1}, \boldsymbol{b}_{K-1}\}$ contains the previous layers of the model. We apply the chain rule to say

$$\begin{aligned}
\frac{\partial \mathcal{L}}{\partial \boldsymbol{\theta}_{K-1}} &= \frac{\partial \mathcal{L}}{\partial \boldsymbol{f}_K} \frac{\partial \boldsymbol{f}_K}{\partial \boldsymbol{\theta}_{K-1}} \\
\frac{\partial \mathcal{L}}{\partial \boldsymbol{\theta}_{K-1}} &= \frac{\partial \mathcal{L}}{\partial \boldsymbol{f}_K} \frac{\partial \boldsymbol{f}_K}{\partial \boldsymbol{f}_{K-1}} \frac{\partial \boldsymbol{f}_{K-1}}{\partial \boldsymbol{\theta}_{K-2}} \\
\frac{\partial \mathcal{L}}{\partial \boldsymbol{\theta}_i} &= \frac{\partial \mathcal{L}}{\partial \boldsymbol{f}_K} \frac{\partial \boldsymbol{f}_K}{\partial \boldsymbol{f}_{K-1}} \cdots \frac{\partial \boldsymbol{f}_{i+2}}{\partial \boldsymbol{f}_{i+1}} \frac{\partial \boldsymbol{f}_{i+1}}{\partial \boldsymbol{\theta}_i}
\end{aligned}$$