1 Softmax

Softmax is the function $\mathbb{R}^n \to \mathbb{R}^n$ given by

$$f_i(x) = \frac{e^{x_i}}{\sum_{k=1}^n e^{x_k}}$$

A bit of algebra shows that its derivatives are given by

$$\frac{\partial f_i}{\partial x_j} = \begin{cases} -\dfrac{e^{x_i} e^{x_j}}{\left(\sum_{k=1}^n e^{x_k}\right)^2} & i \neq j \\ \left(\dfrac{e^{x_i}}{\sum_{k=1}^n e^{x_k}}\right)\left(1 - \dfrac{e^{x_i}}{\sum_{k=1}^n e^{x_k}}\right) & i = j \end{cases}$$

Note that this is symmetric. If we let $y = f(x)$ then this can be rewritten in vectorized form as

$$\frac{\partial f}{\partial x} = \mathrm{diag}(y) - yy^T$$

Computing the product $(\frac{\partial f}{\partial x})z$ for a vector $z \in \mathbb{R}^n$ is given by

$$\left(\frac{\partial f}{\partial x}\right) z = y \circ z - yy^T z = y \circ (z - (y^T z)\mathbf{1})$$

where $\circ$ is an elementwise product and $\mathbf{1} \in \mathbb{R}^n$ is a constant vector of ones.

Now suppose that $X \in \mathbb{R}^{n\times m}$ is a matrix of m inputs stored in columns, and $Y \in \mathbb{R}^{n\times m}$ is the matrix of outputs obtained by applying f to each column of X. Given a matrix $dY \in \mathbb{R}^{n\times m}$ of upstream derivatives, in the backprop step we must compute the matrix $dX \in \mathbb{R}^{n\times m}$ whose ith column is equal to $\frac{\partial f}{\partial x}$ evaluted at the ith column of X, multiplied by the ith column of Y. Using the above results, it is clear that

$$dX = Y \circ (dY - \mathbf{1}\mathbf{1}^T (Y \circ dY))$$

This can be efficiently implemented in numpy using broadcasting as:

```
dX = Y * (dY - np.sum(Y * dY, axis=0))
```