DS-GA 1008: Deep Learning, Spring 2020
Homework Assignment 1

```
He who learns but does not think is lost.
He who thinks but does not learn is in great danger.
Confucius (551 - 479 BC)
```

1. Backprop

Backpropagation or "backward propagation through errors" is a method which calculates the gradient of the loss function of a neural network with respect to its weights.

1.1. Affine Module

The chain rule is at the heart of backpropagation. Suppose we are given $\boldsymbol{x} \in \mathbb{R}^2$ and that we use an affine transformation with parameters $\boldsymbol{W} \in \mathbb{R}^{2\times 2}$ and $\boldsymbol{b} \in \mathbb{R}^2$ defined by:

$$\boldsymbol{y} = \boldsymbol{W}\boldsymbol{x} + \boldsymbol{b}, \tag{1}$$

(a) Suppose an arbitrary cost function $C(\boldsymbol{y})$ and that we are given the gradient $\partial C/\partial \boldsymbol{y}$. Give an expression for $\partial C/\partial \boldsymbol{W}$ and $\partial C/\partial \boldsymbol{b}$ in terms of $\partial C/\partial \boldsymbol{y}$ and $\boldsymbol{x}$ using the chain rule.

Answer:

$$\frac{\partial C}{\partial W} = \frac{\partial C}{\partial y}\frac{\partial y}{\partial W} = \frac{\partial C}{\partial y}x \tag{2}$$

$$\frac{\partial C}{\partial b} = \frac{\partial C}{\partial y}\frac{\partial y}{\partial b} = \frac{\partial C}{\partial y} \tag{3}$$

(b) If we define a new cost $C_2(\boldsymbol{y}) = 3 * C(\boldsymbol{y})$, do we know how our gradients with respect to $\boldsymbol{W}, \boldsymbol{b}$ change without knowing the particular form of $C(\boldsymbol{y})$ or $\partial C/\partial \boldsymbol{y}$?

Answer:

$$\frac{\partial C_2}{\partial W} = 3 * \frac{\partial C}{\partial W} \tag{4}$$

$$\frac{\partial C_2}{\partial b} = 3 * \frac{\partial C}{\partial b} \tag{5}$$

1.2. Softmax Module

The softmax expression is at the crux of multi-class classification. After receiving K unconstrained values in the form of a vector $\boldsymbol{x} \in \mathbb{R}^K$, the softmax function normalizes these values to K positive values that all sum to 1. The softmax is defined as

$$\boldsymbol{y} = \text{softmax}_\beta(\boldsymbol{x}), \quad y_k = \frac{\exp(\beta x_k)}{\sum_n \exp(\beta x_n)}, \tag{6}$$

where $\sum_y^K y_k = 1$, $y_k \geq 0$ for all k, and $\exp(z) = e^z$. We usually let the softmax input $\boldsymbol{x} \in \mathbb{R}^K$ be the output of a preceding module (some feature representation) whose input is a datapoint d. Then we interpret y_k as the probability that datapoint d belongs to class k.

Since this module can be connected to others in backprop using the chain rule, we just need to compute the softmax's gradient in isolation. What is the expression for $\partial y_i / \partial x_j$? (Hint: Answer differs when $i = j$ and $i \neq j$).

Answer:

$$i \neq j : \frac{\partial y_i}{\partial x_j} = -\frac{\exp(\beta x_i)}{(\sum_n \exp(\beta x_n))^2} \beta \exp(\beta x_j) \tag{7}$$

$$i = j : \frac{\partial y_i}{\partial x_i} = \frac{\sum_{n \neq i} \exp(\beta x_n)}{(\sum_n \exp(\beta x_n))^2} \beta \exp(\beta x_i) \tag{8}$$

$$(\frac{\exp(\beta x_i)}{\sum_n \exp(\beta x_n)} = 1 - \frac{\sum_{n \neq i} \exp(\beta x_n)}{\sum_n \exp(\beta x_n)}) \tag{9}$$