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

Due: Tuesday, February 18th, 2020 at 7pm

```
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.
$\frac{\partial C}{\partial \boldsymbol{W}} = \frac{\partial C}{\partial \boldsymbol{y}}\frac{\partial \boldsymbol{y}}{\partial \boldsymbol{W}} = \frac{\partial C}{\partial \boldsymbol{y}}\boldsymbol{x}$
$\frac{\partial C}{\partial \boldsymbol{b}} = \frac{\partial C}{\partial \boldsymbol{y}}\frac{\partial \boldsymbol{y}}{\partial \boldsymbol{b}} = \frac{\partial C}{\partial \boldsymbol{y}} * \mathbb{1} = \frac{\partial C}{\partial \boldsymbol{y}}$

(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 $\partial C/\partial \boldsymbol{y}$?
Yes. Since $C_2(\boldsymbol{y}) = 3 * C(\boldsymbol{y})$, $\frac{\partial C_2}{\partial \boldsymbol{y}} = 3 * \frac{\partial C}{\partial \boldsymbol{y}}$. Therefore, $\frac{\partial C_2}{\partial \boldsymbol{W}}=3*\frac{\partial C}{\partial \boldsymbol{W}}$, $\frac{\partial C_2}{\partial \boldsymbol{b}}=3*\frac{\partial C}{\partial \boldsymbol{b}}$.

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

$$\boldsymbol{y} = \text{softmax}_\beta(\boldsymbol{x}), \quad y_k = \frac{\exp(\beta x_k)}{\sum_n \exp(\beta x_n)}, \tag{2}$$

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.

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

Due: Tuesday, February 18th, 2020 at 7pm

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 $i \neq j$).

$$\frac{\partial y_i}{\partial x_j} = \frac{\partial \frac{e^{\beta x_i}}{\sum_k e^{\beta x_k}}}{\partial x_j}$$

If $i = j$, $\frac{\partial \frac{e^{\beta x_i}}{\sum_k e^{\beta x_k}}}{\partial x_i} = \frac{\beta e^{\beta x_i} \sum_k e^{\beta x_k} - e^{\beta x_i} \beta e^{\beta x_i}}{(\sum_k e^{\beta x_k})^2} = \frac{\beta e^{\beta x_i}}{\sum_k e^{\beta x_k}} \frac{\sum_k e^{\beta x_k} - e^{\beta x_i}}{\sum_k e^{\beta x_k}}$

$= \frac{\beta e^{\beta x_i}}{\sum_k e^{\beta x_k}} (1 - \frac{e^{\beta x_i}}{\sum_k e^{\beta x_k}}) = \beta y_i(1 - y_i)$

If $i \neq j$, $\frac{\partial \frac{e^{\beta x_i}}{\sum_k e^{\beta x_k}}}{\partial x_j} = \frac{0 - e^{\beta x_i} \beta e^{\beta x_j}}{(\sum_k e^{\beta x_k})^2} = -\frac{\beta e^{\beta x_j}}{\sum_k e^{\beta x_k}} \frac{e^{\beta x_i}}{\sum_k e^{\beta x_k}} = -\beta y_j y_i$

2. PyTorch

If you haven't already, install most recent versions of Python (3.6 or higher), PyTorch (we recommend using conda for the installation), and Jupyter.

Complete the programming exercises provided in the homework 1 directory of the course Google Drive folder (link).

3. Evaluation

Homework is worth a total of 100 points.

- Part 1 - 50 points
- Part 2 - 50 points

4. Submission

You are required to write up your solutions to Part 1 using markdown or LaTeX.
Please submit the homework on the NYU classes assignment page. Please upload the following:

- `First-name_Last-name_netID_A1.tex` (or `.md`) file for Part 1
- `First-name_Last-name_netID_A1.pdf` file for Part 1
- `First-name_Last-name_netID_A1_code.ipynb` file for Part 2
- `First-name_Last-name_netID_A1_code.pdf` file for Part 2
  (you can use Jupyter Notebook's "File → Download as → PDF" feature)

5. Disclaimers

You are allowed to discuss problems with other students in the class but have to write up your solutions on your own.
As feedback might be provided during the first days, the current homework assignment might be undergoing some minor changes. We'll notify you if this happens.