DS-GA 1008:
Deep Learning, Spring 2019
Homework Assignment 1
Yves Greatti - yg390

1 Backpropagation

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 Warm-up

The chain rule is at the heart of backpropagation. Assume you are given input $\boldsymbol{x}$ and output $\boldsymbol{y}$, both in $\mathbb{R}^2$, and the error backpropagated to the output is $\frac{\partial L}{\partial y}$. In particular, let

$$\boldsymbol{y} = \boldsymbol{W}\boldsymbol{x} + \boldsymbol{b}$$

where $\boldsymbol{W} \in \mathbb{R}^{2x2}$ and $\boldsymbol{x}, \boldsymbol{b} \in \mathbb{R}^2$. Give an expression for $\frac{\partial L}{\partial \boldsymbol{W}}$ and $\frac{\partial L}{\partial \boldsymbol{b}}$ in terns of $\frac{\partial L}{\partial \boldsymbol{y}}$ and $\boldsymbol{x}$ using the chain rule.

$$\begin{aligned}
\frac{\partial L}{\partial \boldsymbol{W}} &= \frac{\partial \boldsymbol{y}}{\partial \boldsymbol{W}} \cdot \frac{\partial L}{\partial \boldsymbol{y}} \\
&= \sum_{i=1}^{2} \frac{\partial y_i}{\partial \boldsymbol{W}} \left(\frac{\partial L}{\partial \boldsymbol{y}}\right)_i \\
&= \left(\frac{\partial L}{\partial \boldsymbol{y}}\right)_1 \begin{bmatrix} \text{——} \ \boldsymbol{x}^\top \ \text{——} \\ \text{——} \ \boldsymbol{0}^\top \ \text{——} \end{bmatrix} + \left(\frac{\partial L}{\partial \boldsymbol{y}}\right)_2 \begin{bmatrix} \text{——} \ \boldsymbol{0}^\top \ \text{——} \\ \text{——} \ \boldsymbol{x}^\top \ \text{——} \end{bmatrix} \\
&= \begin{bmatrix} \left(\frac{\partial L}{\partial \boldsymbol{y}}\right)_1 \cdot \boldsymbol{x}^\top \\ \left(\frac{\partial L}{\partial \boldsymbol{y}}\right)_2 \cdot \boldsymbol{x}^\top \end{bmatrix} \\
&= \frac{\partial L}{\partial \boldsymbol{y}} \boldsymbol{x}^\top
\end{aligned}$$

We can write this as an outer product:

$$\frac{\partial L}{\partial \boldsymbol{W}}^\top = \frac{\partial L}{\partial \boldsymbol{y}}^\top \otimes \boldsymbol{x}^\top$$

Now we have for $\frac{\partial L}{\partial \boldsymbol{b}}$:

$$\begin{aligned}
\frac{\partial L}{\partial \boldsymbol{b}} &= \frac{\partial \boldsymbol{y}}{\partial \boldsymbol{b}} \cdot \frac{\partial L}{\partial \boldsymbol{y}} \\
&= \sum_{i=1}^{2} \frac{\partial y_i}{\partial \boldsymbol{b}} \left(\frac{\partial L}{\partial \boldsymbol{y}} \right)_i \\
&= 1 \cdot \left(\frac{\partial L}{\partial \boldsymbol{y}} \right)_1 + 1 \cdot \left(\frac{\partial L}{\partial \boldsymbol{y}} \right)_2 \\
&= \sum_{i=1}^{2} \left(\frac{\partial L}{\partial \boldsymbol{y}} \right)_i
\end{aligned}$$

1.2 Softmax

Multinomial logistic regression is a generalization of logistic regression into multiple classes. The softmax expression is at the crux of this technique. After receiving n unconstrained values, the softmax function normalizes these values to n values that all sum to 1. This can then be perceived as probabilities attributed to the various classes by a classifier. Your task here is to back-propagate error through this module. The softmax expression which indicates the probability of the j-th class is as follows:

$$\mathrm{P}(z = j \mid \boldsymbol{x}) = y_j = \frac{\exp(\beta x_j)}{\sum_i \exp(\beta x_i)}$$

What is the expression for $\frac{\partial y_j}{x_i}$? (Hint: Answer differs when i = j and i $\neq$ j) Note that the variables $\boldsymbol{x}$ and $\boldsymbol{y}$ aren't scalars but vectors. While $\boldsymbol{x}$ represents the n values input to the system, $\boldsymbol{y}$ represents the n probabilities output from the system. Therefore, the expression y_j represents the j-th element of $\boldsymbol{y}$.

Solution: From the quotient rule for differential functions, we have:

$$\frac{\partial y_j}{\partial x_i} = \frac{\sum_k \exp(\beta x_k) \cdot \frac{\partial e^{\beta x_j}}{\partial x_i} - e^{\beta x_j} \frac{\partial \sum_k \exp(\beta x_k)}{\partial x_i}}{\left(\sum_k \exp(\beta x_k) \right)^2}$$

if $i = j$ then:

$$\begin{aligned}
\frac{\partial e^{\beta x_j}}{\partial x_i} &= \beta e^{\beta x_i} \\
\frac{\partial \sum_k \exp(\beta x_k)}{\partial x_i} &= \beta e^{\beta x_i}
\end{aligned}$$

Thus

$$\frac{\partial y_i}{\partial x_i} = \frac{\sum_k \exp(\beta x_k) \cdot \beta e^{\beta x_i} - \beta e^{\beta x_i} e^{\beta x_i}}{\left(\sum_k \exp(\beta x_k)\right)^2}$$
$$= \beta \frac{\exp(\beta x_i)}{\sum_k \exp(\beta x_k)} \left(1 - \frac{\exp(\beta x_i)}{\sum_k \exp(\beta x_k)}\right)$$

if $i \neq j$: $\frac{\partial e^{\beta x_j}}{\partial x_i} = 0$ then:

$$\frac{\partial y_j}{\partial x_i} = -\frac{\beta e^{\beta x_i} e^{\beta x_j}}{\left(\sum_k \exp(\beta x_k)\right)^2}$$

So the backpropagation of the error through softmax is:

$$\frac{\partial y_j}{\partial x_i} = \begin{cases} \beta\ p_i\ (1 - p_i) & \text{if } i = j \\ -\beta\ p_i\ p_j & \text{otherwise} \end{cases} \quad \text{where } p_j = y_j = \frac{\exp(\beta x_j)}{\sum_i \exp(\beta x_i)} \tag{1}$$

Or using Kronecker delta δ_{ij}:

$$\frac{\partial y_j}{\partial x_i} = \beta\ p_i\ (\delta_{ij} - p_j) \tag{2}$$