Deep Neural Networks

Assignment 8

Assignment due by: 18.12.2019, Discussions on: 08.01.2020

Question 1 Backpropagation (12 points)

In this assignment we will go back to native Python to manually implement Backpropagation. We work with the MNIST dataset again and we use a neural network with a single hidden layer. The structure will be the following (all vectors are row vectors):

$$\mathcal{L} = H(\boldsymbol{y}', \boldsymbol{y}),$$

$$\begin{aligned}\boldsymbol{y} &= \mathrm{Softmax}(\boldsymbol{u}^{(2)}),\\ \boldsymbol{u}^{(2)} &= \boldsymbol{z}^{(1)}\boldsymbol{W}^{(2)} + \boldsymbol{b}^{(2)},\\ \boldsymbol{z}^{(1)} &= \sigma(\boldsymbol{u}^{(1)}),\\ \boldsymbol{u}^{(1)} &= \boldsymbol{x}\boldsymbol{W}^{(1)} + \boldsymbol{b}^{(1)},\end{aligned}$$

where $\sigma(x)$ is the logistic sigmoid function. The derivative of the Softmax function has already been discussed in assignment 4, so in this exercise we will focus on the rest of the backpropagation process. Thus you can use $\frac{\partial \mathcal{L}}{\partial u_i^{(2)}}$ without expanding it further.

To perform gradient descent we need the gradients, i.e. the derivatives w.r.t. the weights and biases, i.e. $\nabla_{\boldsymbol{W}^{(i)}}\mathcal{L}$ and $\nabla_{\boldsymbol{b}^{(i)}}\mathcal{L}$ for $i \in \{1, 2\}$.

Remember that the chain rule is $\frac{\partial f}{\partial x} = \sum_i \frac{\partial f}{\partial z_i}\frac{\partial z_i}{\partial x}$ when the intermediate variable $\boldsymbol{z}$ is a vector. For this question you *don't* need to take into account that you are processing multiple inputs (a batch) at the same time. It might be helpful to solve this exercise by using the index method (show it for the i, j-th entry).

(a) Just using the chain rule, expand $\nabla_{\boldsymbol{W}^{(i)}}\mathcal{L}$ and $\nabla_{\boldsymbol{b}^{(i)}}\mathcal{L}$, $i = 1, 2$, into a chain of partial derivatives (without calculating the derivatives, just write down the correct derivative expressions). (3 points)

(b) Calculate the derivatives and write out the resulting formulas (Except $\frac{\partial \mathcal{L}}{\partial u_i^{(2)}}$). You are allowed to abbreviate the derivative of $\sigma(x)$ as $\sigma'(x)$. (6 points)

(c) Find common subexpressions that can be used to avoid duplicate calculations. Write down the order of calculations to compute all the gradients efficiently. (3 points)

Hint: If you are unsure whether your results are correct, check whether the dimensions of the resulting vectors and matrices match up correctly.

Question 2 Implementing Backpropagation (8 points)

The file `backprop.py` contains most of the code for training a 1-hidden-layer neural network on the MNIST dataset. Your task is to add the appropriate operations to calculate the partial derivatives and the gradients that are then used in gradient descent. Make sure you avoid computing things twice by pre-computing the common subexpressions.

For this task you need to take into account that the batches consist of multiple images, so the input vector $\boldsymbol{x}$ is extended to a design matrix over a mini batch and similarly the other vectors. This means that you have to average over the batch at the end to compute the gradient.