Machine Learning Assignment 2: Neural Networks

Deadline: Thursday 23rd November 2023

Please submit your solution in PDF format (preferably, but not necessarily, LaTeX—this .tex file can be found on iCorsi). Handwriting and scanned documents are not allowed. In case you need further help, please write on iCorsi or contact me at vincent.herrmann@usi.ch.

Question 1 (20 points). A two-layer neural network is defined by the following equation:

$$y_k := \mathbf{w}^{(2)} f\left(\mathbf{x}_k W^{(1)} + \mathbf{b}^{(1)}\right)^T + b^{(2)}$$

The values of the input vectors $\mathbf{x}_k$ (summarized in the matrix X), targets t_k (summarized in the vector $\mathbf{t}$), weights $W^{(1)}$ and $\mathbf{w}^{(2)}$ as well as biases $\mathbf{b}^{(1)}$ and $b^{(2)}$ are given below.

$$\begin{aligned}
X &= \begin{bmatrix} — & \mathbf{x}_1 & — \\ — & \mathbf{x}_2 & — \\ — & \mathbf{x}_2 & — \\ — & \mathbf{x}_2 & — \end{bmatrix} = [[0.6, -1.0], [0.8, -1.0], [-0.4, 0.9], [0.2, 0.0]] \\
\mathbf{t} &= \begin{bmatrix} t_1 \\ t_2 \\ t_3 \\ t_4 \end{bmatrix} = [-0.8, -0.1, 0.9, 0.7] \\
W^{(1)} &= [[-0.8, -0.7, 0.6], [-1.0, 0.5, -1.0]] \\
\mathbf{b}^{(1)} &= [-0.2, -1.0, -0.7] \\
\mathbf{w}^{(2)} &= [0.1, -1.0, 0.5] \\
b^{(2)} &= -0.7
\end{aligned}$$

The function f is the ReLU nonlinearity. The loss of the model is

$$L := \frac{1}{2} \sum_{k=1}^{4} (y_k - t_k)^2.$$

Numerically calculate the gradients of L with respect to $W^{(1)}$, $\mathbf{w}^{(2)}$, $\mathbf{b}^{(1)}$ and $b^{(2)}$, and explain your process. You may use a calculator or math software (numpy, matlab, ...), but no auto-differentiation libraries like PyTorch or Jax. Tip: If you use matrix-matrix multiplications involving X, you don't need to do multiple explicit forward and backward passes.

Question 2 (15 points). As mentioned in the lecture and on the slides, often the Softmax nonlinearity at the output neurons and the Crossentropy loss are combined into a single Softmax+Crossentropy loss function. With this loss function, the error is $E = -\sum_k t_k \log\left(\frac{\exp(y_k)}{\sum_i \exp(y_i)}\right)$, with y_k being the inputs to the softmax function (i.e. the outputs of the neural network), and t_k being the elements of the target vector. Derive and simplify the gradient $\frac{\partial E}{\partial y_i}$. How can it be simplified even more if the target is a one-hot vector (one element has the value 1, all the others are zero)?

Question 3 (15 points). If we choose a learning rate that is too high, gradient descent can diverge (even in the case of convex functions). Assume we want to use gradient descent to find the value of x that minimizes $f(x) = 5x^2 + 2$. The update via gradient descent can be written as $x_{n+1} = x_n - \eta f'(x_n)$, with η being the learning rate. What is the range of values that η can take so that gradient descent converges to the minimum from any starting point? Provide an explanation of your result (can be visual).