Advanced Computer Vision

Exercise Sheet 3

Winter Term 2023
Prof. Dr. Niels Landwehr
Dr. Ujjwal

Available: 21.11.2023
Hand in until: 28.11.2023 at 11:59 AM
Exercise session: 01.12.2023

Task 1 – Artificial Neural Network [15 points]

Consider a simple artificial neural network (ANN):

$$\begin{aligned} f_{\boldsymbol{\theta}} :\ & \mathbb{R} \to \mathbb{R} \\ x \ \mapsto\ & \mathbf{W}_2\, \sigma(\mathbf{W}_1 x + \mathbf{b}_1) + b_2\,. \end{aligned}$$

It consists of one hidden layer $\mathbf{W}_1 \in \mathbb{R}^{k\times 1}$, one output layer $\mathbf{W}_2 \in \mathbb{R}^{1\times k}$ and the two biases $\mathbf{b}_1 \in \mathbb{R}^k$ and $b_2 \in \mathbb{R}$. The activation function σ is given by the sigmoid function

$$\begin{aligned} \sigma :\ & \mathbb{R} \to \mathbb{R} \\ t \ \mapsto\ & \frac{1}{1+e^{-t}}. \end{aligned}$$

When writing $\sigma(\mathbf{W})$ for a matrix or vector $\mathbf{W}$, we mean that the function σ is applied element-wise to $\mathbf{W}$ (i.e., on every entry in the vector or matrix).

The given training data are $\{(x_1, y_1), \ldots, (x_n, y_n)\}$, $x_i,\ y_i \in \mathbb{R}$. The loss for the i-th sample pair is given by the mean squared error

$$\ell(f_{\boldsymbol{\theta}}(x_i), y_i) = \left(f_{\boldsymbol{\theta}}(x_i) - y_i\right)^2\ .$$

Hence, the setting is a nonlinear regression task.

Show for $k = 2$ that the partial derivatives $\frac{\partial \ell}{\partial \mathbf{W}_1}$, $\frac{\partial \ell}{\partial \mathbf{W}_2}$, $\frac{\partial \ell}{\partial \mathbf{b}_1}$, $\frac{\partial \ell}{\partial b_2}$ of the loss function are given by:

$$\begin{aligned} \frac{\partial \ell}{\partial \mathbf{W}_1} &= 2\,[f(x) - y]\, \mathbf{W}_2^T \circ \sigma'(\mathbf{W}_1 x + \mathbf{b}_1)\, x \\ \frac{\partial \ell}{\partial \mathbf{W}_2} &= 2\,[f(x) - y]\ (\sigma(\mathbf{W}_1 x + \mathbf{b}_1))^T \\ \frac{\partial \ell}{\partial \mathbf{b}_1} &= 2\,[f(x) - y]\, \mathbf{W}_2^T \circ \sigma'(\mathbf{W}_1 x + \mathbf{b}_1) \\ \frac{\partial \ell}{\partial b_2} &= 2\,[f(x) - y], \end{aligned}$$

where $\mathbf{W}_i^T$ is the transposed of matrix $\mathbf{W}_i$ and $\mathbf{W} \circ \mathbf{V}$ denotes the element-wise product of $\mathbf{W}$ and $\mathbf{V}$ (the so-called Hadamard product $\circ$). Again, $\sigma'(\mathbf{W})$ denotes element-wise application of the derivative of σ to a matrix or vector $\mathbf{W}$. Remark: the derivative of $\sigma(t) = \frac{1}{1+e^{-t}}$ is $\sigma'(t) = \sigma(t)(1 - \sigma(t))$.

Hint: write the matrices explicitly, that is

$$\mathbf{W}_1 = \begin{pmatrix} W_1^{(1)} \\ W_1^{(2)} \end{pmatrix},\quad \mathbf{b}_1 = \begin{pmatrix} b_1^{(1)} \\ b_1^{(2)} \end{pmatrix},\quad \mathbf{W}_2 = (W_2^{(1)}\ W_2^{(2)}).$$

Task 2 – Implement a Simple Neural Network [15 points]

Write a Python notebook that implements the simple neural network from **Task 1**. The ANN has one hidden layer with k nodes and 1 node in the output layer. Your neural network has to solve a regression problem ('curve fitting') using the quadratic loss as objective function. Generate a set of data points $\{(x_1, y_1), ..., (x_n, y_n)\}$ with uniformly distributed random values $x_i \in [-1, +1]$ and with labels $y_i = f_{\text{noisy}}(x_i)$, where $f_{\text{noisy}}(x)$ is the Gauss-shaped function $f_{\text{org}}(x)$ with additional normal distributed noise:

$$f_{\text{org}}(x) = \frac{1}{1 + \exp(-10\,x^2)} - \frac{1}{2}$$
$$f_{\text{noisy}}(x) = f_{\text{org}}(x) + s\, f_{\text{normal}}(x)\,.$$

Implement a gradient descent algorithm for your neural network based on the derivatives from Task 1. That is, compute the derivates in your code and update the weights by following the negative gradient multiplied with an appropriate learning rate. Initialize the weights in your network randomly, e.g. by drawing each weight from a uniform distribution centered at zero.
Plot the loss function over the gradient iterations and plot the data points and the function your neural net fitted to the data points. Try how different numbers of nodes of the hidden layer influence the resulting function fit. You could also play with different functions for $f_{\text{org}}(x)$.

Task 3 – Autodiff [20 points]

Write down and draw the graph of primitive operations for the function

$$f(x_0, x_1, w_0, w_1, w_2) = \left[\tanh\left(\frac{1}{w_0 x_0 + w_1 x_1 + w_2}\right) - 1\right]^2, \qquad x_0, x_1, w_0, w_1, w_2 \in \mathbb{R}$$

as shown in the lecture. Write down all local gradients and carry out the forward- and backward pass as shown in the lecture at the point $x_0 = 1.2$, $x_1 = 3.1$, $w_0 = 1.5$, $w_1 = -1.5$, $w_2 = 2.0$. Use a precision of at least 3 digits after the comma. For the backward pass, show that $\frac{d}{dx}\tanh(x) = 1 - \tanh(x)^2$.

If you like, you can calculate the gradient analytically (recommended to use a software tool) and compare the result of your backward pass with the analytical gradient at the given point.