Maths for Intelligent Systems
Exercise 5

Marc Toussaint

Machine Learning & Robotics lab, U Stuttgart

Universitätsstraße 38, 70569 Stuttgart, Germany

November 28, 2018

1 Backprop in a Neural Net

We consider again the function

$$f: \mathbb{R}^{h_0} \to \mathbb{R}^{h_3} \ , \quad f(x_0) = W_2 \sigma(W_1 \sigma(W_0 x_0)) \ ,$$

where $W_l \in \mathbb{R}^{h_{l+1} \times h_l}$ and $\sigma : \mathbb{R} \to \mathbb{R}$ is a differentiable activation function which is applied element-wise. We established last time that

$$\frac{df}{dx_0} = \frac{\partial f}{\partial x_2} \frac{\partial x_2}{\partial z_2} \frac{\partial z_2}{\partial x_1} \frac{\partial x_1}{\partial z_1} \frac{\partial z_1}{\partial x_0}$$

with:

$$\frac{\partial x_l}{\partial z_l} = \text{diag}(x_l \circ (1 - x_l)) \ , \quad \frac{\partial z_{l+1}}{\partial x_l} = W_l \ , \quad \frac{\partial f}{\partial x_2} = W_2$$

Note: In the following we still let f be a h_3-dimensional vector. For those that are confused with the resulting tensors, simplify to f being a single scalar output.

(a) Derive also the necessary equations to get the derivative w.r.t. the weight matrices W_l, that is the Jacobian tensor

$$\frac{df}{dW_l}$$

(b) Write code to implement $f(x)$ and $\frac{df}{dx_0}$ and $\frac{df}{dW_l}$.

To test this, choose layer sizes $(h_0, h_1, h_2, h_3) = (2, 10, 10, 2)$, i.e., 2 input and 2 output dimensions, and hidden layers of dimension 10.

For testing, choose random inputs sampled from $x \sim$`randn(2,1)`

And choose random weight matrices $W_l \sim \frac{1}{\sqrt{h_{l+1}}}$`rand(h[l+1],h[l])`.

Check the implemented Jacobian by comparing to the finite difference approximation.

Debugging Tip: If your first try does not work right away, the typical approach to debug is to "comment out" parts of your function f and df. For instance, start with testing $f(x) = W_0 x_0$; then test $f(x) = \sigma(W_0 x_0)$; then $f(x) = W_1 \sigma(W_0 x_0)$; then I'm sure all bugs are found.

(c) Bonus: Try to train the network to become the identity mapping. In the simplest case, use "stochastic gradient descent", meaning that you sample an input, compute the gradients $w_l = \frac{d(f(x)-x)^2}{dW_l}$, and make tiny updates $W_l \leftarrow W_l - \alpha w_l$.

2 Logistic Regression Gradient & Hessian

Consider the function

$$L:\ \mathbb{R}^d \to \mathbb{R}:\ L(\beta) = -\sum_{i=1}^n \Big[y_i \log \sigma(x_i^\top \beta) + (1-y_i)\log[1-\sigma(x_i^\top \beta)]\Big] \ +\ \lambda \beta^\top \beta\ ,$$

where $x_i \in \mathbb{R}^d$ is the ith row of a matrix $X \in \mathbb{R}^{n\times d}$, and $y \in \{0,1\}^n$ is a vector of 0s and 1s only. Here, $\sigma(z) = 1/(e^{-z}+1)$ is the sigmoid function, with $\sigma'(z) = \sigma(z)(1-\sigma(z))$.

Derive the gradient $\frac{\partial}{\partial \beta} L(\beta)$, as well as the Hessian

$$\nabla^2 L(\beta) = \frac{\partial^2}{\partial \beta^2} L(\beta)\ .$$