Maths for Intelligent Systems
Exercise 4

Marc Toussaint, Andrea Baisero
Machine Learning & Robotics lab, U Stuttgart
Universitätsstraße 38, 70569 Stuttgart, Germany

November 22, 2018

1 Multivariate Calculus

Given tensors $y \in \mathbb{R}^{a\times\dots\times z}$ and $x \in \mathbb{R}^{\alpha\times\dots\times\omega}$ where y is a function of x, the Jacobian tensor $J = \partial_x y$ is in $\mathbb{R}^{a\times\dots\times z\times\alpha\times\dots\times\omega}$ and has coefficients

$$J_{i,j,k,\dots,l,m,n\dots} = \frac{\partial}{\partial x_{l,m,n,\dots}} y_{i,j,k,\dots}$$

(All "output" indices come before all "input" indices".)
Compute the following Jacobian tensors

(a) $\frac{\partial}{\partial x} x$, where x is a vector

(b) $\frac{\partial}{\partial x} x^\top A x$, where A is a matrix

(c) $\frac{\partial}{\partial A} y^\top A x$, where x and y are vectors (note, we take the derivative w.r.t. A)

(d) $\frac{\partial}{\partial A} A x$

(e) $\frac{\partial}{\partial x} f(x)^\top A g(x)$, where f and g are vector-valued functions

2 Finite Difference Gradient Checking

The following exercises will require to code basic functions and derivatives. You can code in your prefereed language (Matlab, NumPy, Julia, whatever).

(a) Implement the following pseudo code for empirical gradient checking in the programming language of your choice:

Input: $x \in \mathbb{R}^n$, function $f : \mathbb{R}^n \to \mathbb{R}^d$, function $df : \mathbb{R}^n \to \mathbb{R}^{d\times n}$

1: initialize $\widehat{J} \in \mathbb{R}^{d\times n}$, and $\epsilon = 10^{-6}$
2: **for** $i = 1 : n$ **do**
3: $\quad \widehat{J}_{\cdot i} = [f(x + \epsilon e_i) - f(x - \epsilon e_i)]/2\epsilon$ *// assigns the ith column of $\widehat{J}$*
4: **end for**
5: if $\|\widehat{J} - df(x)\|_\infty < 10^{-4}$ return true; else false

---

Here e_i is the ith standard basis vector in $\mathbb{R}^n$.

(b) Test this for

- $f : x \mapsto Ax$, $df : x \mapsto A$, where you sample $x \sim$`randn(n,1)` ($\mathcal{N}(0,1)$ in Matlab) and $A \sim$`randn(m,n)`
- $f : x \mapsto x^\top x$, $df : x \mapsto 2x^\top$, where $x \sim$`randn(n,1)`

3 Backprop in a Neural Net

Consider the function

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

where $W_i \in \mathbb{R}^{h_{i+1}\times h_i}$ and $\sigma : \mathbb{R} \to \mathbb{R}$ is a differentiable activation function. We also describe the function as the computation graph:

$$x_0 \quad\mapsto\quad z_1 = W_0 x_0 \quad\mapsto\quad x_1 = \sigma(z_1) \quad\mapsto\quad z_2 = W_1 x_1 \quad\mapsto\quad x_2 = \sigma(z_2) \quad\mapsto\quad f = W_2 x_2$$

Derive pseudo code to efficiently compute $\frac{df}{dx_0}$. (Ideally also for deeper networks.)