Anna Krasilnikova 2562668
Mirza Misbah Mubeen Baig 2571567
Shahzain Mehboob 2571564

Exercise 3.1 – *Vector Derivatives*

In this lecture we will often encounter functions of several variables, i.e. $\mathbb{R}^n \longrightarrow \mathbb{R}$. Knowing how to compute the derivatives of such functions will prove helpful for understanding formulas throughout this lecture. Now let $f: \mathbb{R}^n \longrightarrow \mathbb{R}, w \in \mathbb{R}^n, A \in \mathbb{R}^{n\times n}$ and $B \in \mathbb{R}^{m\times n}$. Prove that the following rules hold:

a) $f(x) = \langle w, x\rangle,$ $then$ $\nabla_x f(x) = w$

Let $w = \begin{matrix} w_1 \\ w_2 \\ \vdots \\ w_n \end{matrix}$, where $w \in \mathbb{R}^n$ and $x = \begin{matrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{matrix}$, where $x \in \mathbb{R}^n$. Then we can write:

$$\langle w, x\rangle = \sum_{i=0}^{n} x_i w_i = x_1 w_1 + x_2 w_2 + \cdots + x_n w_n = x^T w$$

$$\langle w, x\rangle = \sum_{i=0}^{n} w_i x_i = w_1 x_1 + w_2 x_2 + \cdots + w_n x_n = w^T x$$

The derivative of $w^T x$ with the respect to x:

$$\frac{\partial \sum_{i=1}^{n} w_i x_i}{\partial x} = w_i \implies \frac{\partial f(x)}{\partial x} = w_1 + \ w_2 + \cdots + w_n$$

We know that $w^T = w$, therefore $\boldsymbol{\nabla_x f(x) = w}$

b) $f(x) = \langle x, Ax\rangle = x^T Ax,$ $then$ $\nabla_x f(x) = Ax + A^T x$

Let $A \in \mathbb{R}^{n\times n}$ and $x \in \mathbb{R}^n$.

$$\frac{\partial x^T Ax}{\partial x} = \frac{\partial x^T A\bar{x}}{\partial x} = \frac{\partial \bar{x}^T A}{\partial x}$$

$\bar{x}$ is the constant term when taking derivative. To compute the derivatives we know from part a) that $\nabla_x f(x) w^T x = w^T$ therefore by substituting $u_1 = A\bar{x}$ and $u_2^T = \bar{x}^T A$.

$$\frac{\partial x^T Ax}{\partial x} = \frac{\partial x^T A\bar{x}}{\partial x} + \frac{\partial \bar{x}^T Ax}{\partial x}$$

$$= \frac{\partial x^T u_1}{\partial x} + \frac{\partial u_2^T x}{\partial x} = u_1^T + u_2^T = (Ax)^T + (x^T A)^T = x^T A^T + A^T x$$

We know that A is a symmetric matrix, therefore we can write

$\boldsymbol{x^T Ax = Ax + A^T x}$, which concludes the proof.

c) $f(x) = \|Bx\|_2^2,$ $then$ $\nabla_x f(x) = 2B^T Bx$

$$\|Bx\|_2^2 = (Bx)^T Bx = (Bx)^T Bx = x^T B^T Bx$$

Taking the derivative w.r.t. x, we get:

$$\boldsymbol{\nabla_x f(x) = 2B^T Bx}$$

d) $f(x) = \|Bx - c\|_2^2,$ $then$ $\nabla_x f(x) = 2B^T(Bx - c)$

$$\|Bx - c\|_2^2 = (Bx - c)^T (Bx - c) = x^T B^T B - c^T Bx - x^T B^T c + c^T c$$

Taking derivative, we get:

$$\nabla_x f(x) = 2B^T Bx - 2B^T c = \boldsymbol{2B^T(Bx - c)}$$

Exercise 3.2 – Computational issues with Softmax

The softmax function plays an important role in neural networks, especially when performing classification. It is defined as follows:

Anna Krasilnikova 2562668
Mirza Misbah Mubeen Baig 2571567
Shahzain Mehboob 2571564

$$softmax: \mathbb{R}^n \longrightarrow \mathbb{R}^k, so\ softmax(x)_i = \frac{\exp(x_i)}{\sum_{j=1}^{n}\exp(x_j)}, i = 1,..k$$

a) *Numerical issues might occur when computing softmax functions o computer. Name these numerical issues and explain them.*
There exists the number of stability issues occurring while computing softmax function. The main issue is potential overflow. Essentially it means following: some values of the sum of $\exp(x_i)$ could be too large to fit in the cell used in computer processor. For example, numpy returns positive infinity for overflowing e^{1000}, thus in actual softmax numerator and denominator become infinity, and numpy returns NAN for the division.
Coming from the overflow – is the opposite, underflow, which can bring zero in the final result of the softmax. The problem there is that in the resulting formula is still $\exp(x_i)$, which might have small value and might cause overflow.

b) *Suggest a remedy to overcome these numerical issues occurring with Softmax computation and explain why it prevent them.*
As detailed in Deep Learning textbook by Ian Goodfellow, Yoshua Bengio and Aaron Courville, chapter 4, a solution to overflow problem is to calculate the softmax value as the following, with $x_M = \max(x_1, x_{2,}, \dots, x_n)$:

$$softmax(x_i) = \frac{exp(x_i - x_M)}{\sum_{i=1}^{n}\exp(x_i - x_M)}$$

This solution doesn't reduce precision (the formulation with and without x_M are mathematically equivalent) and is able to avoid overflow.
The same rule can be applied to avoid underflow problem.

c) *Show that for any input $x \in \mathbb{R}^n$, $softmax(x)_i \geq 0$ and $\sum_{i=1}^{n} softmax(x)_i = 1$. That is, the vector $softmax(x) \in \mathbb{R}^k$ can be understood as a probability distribution.*

We know that $softmax(x)_i = \frac{\exp(x_i)}{\sum_{j=1}^{n}\exp(x_j)}$, and that $\exp(x) > 0\ for\ any\ x$, hence it's clear that $\frac{\exp(x_i)}{\sum_{j=1}^{n}\exp(x_j)}$ cannot ever be greater than 1.

Formally, we can show it like this:

Given that $\exp(x_j) = \exp(x_i), if\ i = j$,

$\sum_{i=1}^{n}\frac{\exp(x_i)}{\sum\exp(x_j)} = \frac{1}{\sum\exp(x_j)}\sum_{i=1}^{n}\exp(x_i) = 1$, this is greater than 0, which concludes the prove.

d) *Compute the first derivative of $softmax(x)$, i.e. compute the Jacobian Matrix of $softmax(x)$.*
Critical

1. Let's start by computing the Jacobian Matrix.

We can apply the following rule of differentiation:

$$y = \frac{f(x)}{g(x)} \Longrightarrow \frac{dy}{dx} = \frac{(f'g - g'f)}{g^2}$$

We can set $f(x) = \exp(x_i)$, $g(x) = \sum\exp(x_j)$:

Then:

$$g'(x) = \exp(x_j), for\ any\ i \in \mathbb{N}$$

$$f'(x) = \begin{cases} \exp(x_i), if\ \frac{de^{x_i}}{dx}, i = j \\ 0, otherwise \end{cases}$$

Anna Krasilnikova 2562668
Mirza Misbah Mubeen Baig 2571567
Shahzain Mehboob 2571564

Now we can plug those values to the actual formula:

$$\frac{\exp(x_i)\sum_{j=1}^{n}\exp(x_j) - \exp(x_j)\exp(x_i)}{(\sum_{j=1}^{n}\exp(x_j))^2} = \begin{cases} \dfrac{\exp(x_i)\sum_{j=1}^{n}\exp(x_j) - \exp(2x_i)}{(\sum_{j=1}^{n}\exp(x_j))^2}, for\ i = j \\ \dfrac{-\exp(x_j)\exp(x_i)}{(\sum_{j=1}^{n}\exp(x_j))^2}, otherwise \end{cases}$$

2. Now we can compose the Jacobian of softmax:

$$\frac{1}{(\sum_{j=1}^{n}\exp(x_j))^2}\begin{bmatrix} \exp(x_1)\sum\exp(x_j) - \exp(2x_1) & \cdots & \cdots & \cdots & -\exp(x_1)\exp(x_n) \\ \vdots & \ddots & & & \vdots \\ \vdots & & \ddots & & \vdots \\ \vdots & & & \ddots & \vdots \\ -\exp(x_1 + x_n) & \cdots & \cdots & \cdots & \exp(x_n)\sum\exp(x_j) - \exp(2x_n) \end{bmatrix}$$

Exercise 2.3 – *Bad Step Size*

Construct a smooth (i.e. continuously differentiable) function $f: \mathbb{R} \longrightarrow \mathbb{R}$, a starting point $x_0 \in \mathbb{R}$ with $f'(0) \neq 0$, and step sizes ϵ_k such that $f(x_k)$ will converge to a local maximum when applying gradient descent method.

To find local minimum of any function we can use gradient descent method. The formula for this method is:

$x_i + 1 = x_i - \varepsilon f'(x_i)$ (1)

Now we have to choose arbitrary function. The function we are choosing here, is:

$$f(n) = \sin(n)$$

Taking the derivative of the function w. r. t. x.

$$f'(n) = \cos(n)$$

To make the algorithm work, we have to start with an initial guess.

Here we are choosing starting point $x_o = 2\pi$ putting the value of x_0 in derivative of the function.

$$f'(x_0) = \cos(2\pi)$$
$$f'(x_0) = 1$$

and choosing epsilon as $\frac{3}{2}\pi$.

As $f(x_0) = \sin(2\pi) \Longrightarrow f(x_0) = 0$, which is showing that initially there is no local maxima.

Now calculating the values for further x by using equation (1):

For x_1:

$$x_1 = x_0 - \varepsilon f'(x_0)$$
$$x_1 = 2\pi - \frac{3}{2}\pi \cdot \cos(2\pi)$$
$$x_1 = \frac{1}{2}\pi$$

For x_2:

$$x_2 = x_1 - \varepsilon f'(x_1)$$
$$x_2 = \frac{1}{2}\pi - \frac{3}{2}\pi \cdot \cos(\frac{1}{2}\pi)$$
$$x_2 = \frac{1}{2}\pi$$

Now as $x_1 = x_2$ which means that all the other x values will be the same.

$$x_1 = x_2 = x_3 \ldots = x_n$$
$$x_0 \neq x;\ \ x \in \{x_1, x_2, \ldots, x_n\}$$

Anna Krasilnikova 2562668
Mirza Misbah Mubeen Baig 2571567
Shahzain Mehboob 2571564

Therefore the function is converging to local maxima when applying gradient descent at $\frac{1}{2}\pi$ with the step size $\frac{3}{2}\pi$ for the function $f(x) = \sin(x)$, using the starting point $x_0 = 2\pi$