Exercise Sheet 5

Zena Al Khalili (7009151)
Sangeet Sagar (7009050
{zeal00001,sasa00001}@stud.uni-saarland.de

December 15, 2020

Exercise 5.1 - Computing Jacobian and Hessian

Jacobian is given by:

$$\mathbf{J} = \begin{bmatrix} \frac{\partial f}{\partial x} & \frac{\partial f}{\partial y} \end{bmatrix}$$
$$\mathbf{J} = \begin{bmatrix} 6xy + 12y^4x^2 - 63x^8y^4 & 3x^2 + 16x^3y^3 - 28x^9y^3 \end{bmatrix}$$

Hessian is given by:

$$\mathbf{H}f = \begin{bmatrix} \frac{\partial^2 f}{\partial x^2} & \frac{\partial^2 u_1}{\partial y \partial x} \\ \frac{\partial^2 f}{\partial x \partial y} & \frac{\partial^2 u_1}{\partial y^2} \end{bmatrix}$$
$$\mathbf{H}f = \begin{bmatrix} 6y + 24xy^4 - 504x^7y^4 & 6x + 48x^2y^3 - 252x^8y^3 \\ 6x + 48x^2y^3 - 252x^8y^3 & 48x^3y^2 - 84x^9y^2 \end{bmatrix}$$

Exercise 5.2 - Taylor Series and Newton's Method

(a)

Given: $f(x) = cos(x)$; $x_o = 0$
Taylor series for $f(x)$ is given by:

$$\implies f(a) + \frac{f'(a)}{1!}(x-a) + \frac{f''(a)}{2!}(x-a)^2 + \frac{f'''(a)}{3!}(x-a)^3 + \cdots$$
$$\implies cos(0) + \frac{-sin(0)}{1!}(x-0) + \frac{-cos(0)}{2!}(x-0)^2 + \frac{sin(0)}{3!}(x-0)^3 + \frac{-cos(0)}{4!}(x-0)^4 + \cdots$$
$$\implies 1 - \frac{x^2}{2!} + \frac{x^4}{4!}$$

General expression for Taylor for $f(x) = cos(x)$ at $x = 0$ is given by:

$$\sum_{n=0}^{\infty} (-1)^n \frac{x^{2n}}{(2n)!}$$

(b)

For this example, the quadratic function will produce a critical point after one iteration, and it's a saddle point.

(c)

Gradient descent method maximizes a function using the knowledge of its first derivative. Newton's method which is a root finding algorithm uses second derivative to maximize a function.

(d)

It is not possible to apply newtons method when:

1. $\mathbf{H}f$ results in an invertible matrix. or for high dimensionality problems when the Hessian Matrix is so big, it will be so computationally so expensive to inverse the Hessian
2. the derivative $f^{'}(x)$ is 0

Exercise 5.3 - Activation Functions

(a)

The derivative of Sigmoid function is:

$$\sigma'(x) = \frac{e^{-x}}{(1+e^{-x})^2} = \frac{1}{1+e^{-x}} . \frac{e^{-x}+1-1}{1+e^{-x}}$$
$$= \frac{1}{1+e^{-x}} . (1 - \frac{1}{1+e^{-x}})$$
$$\sigma'(x) = \sigma(x).(1-\sigma(x))$$

The derivative of hyperbolic tan is:

$$tanh'(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}} = \frac{(e^x+e^{-x})^2 - (e^x - e^{-x})^2}{(e^x+e^{-x})^2}$$

$$= 1 - (\frac{e^x - e^{-x}}{e^x + e^{-x}})^2$$

The derivative of ReLU is:

$$ReLU'(x) = \begin{cases} 0 & x < 0 \\ 1 & x > 0 \end{cases}$$

(b)

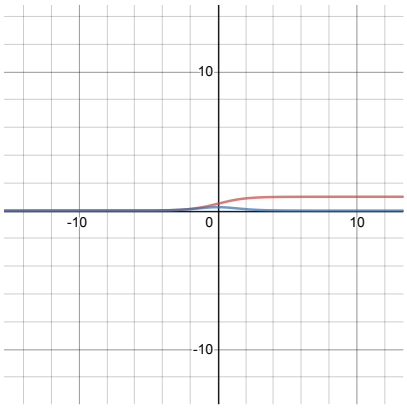

Figure 1: $\sigma(x)$

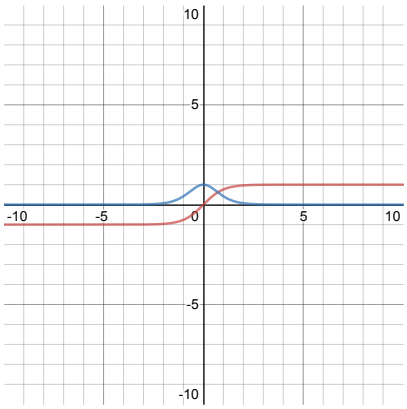

Figure 2: $tanh(x)$

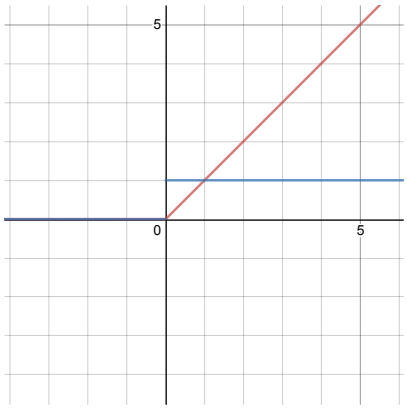

Figure 3: $ReLU(x)$

For each figure of the above the red graph indicates the function and the blue one indicates the derivative.

The Sigmoid function produces values between 0-1, so it's better be used to output probabilities or binary classification. The disadvantage of Sigmoid is when the x value is small or big the gradient is 0, as shown in the figure above, and there's no learning.

The hyperbolic tan will map values to be between -1 and 1, so the output is centered at 0. Usually used in hidden layers of a neural network as it's values lies between-1 to 1 ,this makes learning for the next layer much easier. The disadvantages, that it also will give gradient to be 0 for some values x and there will be

no learning.

ReLu is less computationally expensive than tanh and sigmoid because it involves simpler mathematical operations. At a time only a few neurons are activated making the network sparse making it efficient and easy for computation. The disadvantage of ReLU is that If the units are not activated initially, then they are always in the off-state as zero gradients flow through them (Dead Neurons).

(c)

The best activation function for binary classification is the Sigmoid function. For multicalss classification it's better to use softmax function as it generate probabilities of classes that sums to 1, which means that classes probabilities are dependent. While Sigmoid will produce independent probabilities that they won't sum to one.