Project 2

Edvard B. Rørnes[*] and Isak O. Rukan[†]
Institute of Physics, University of Oslo,
0371 Oslo, Norway
(Dated: October 27, 2024)

We build a simple neural network to identify malignant tumors using breast cancer data. Different activation functions such as: Sigmoid, tanh, ReLU and LeakyReLU were used. The number of epochs and hidden nodes along with values for the hyper-parameter λ and the learning rate η were analyzed and tuned to optimal values. The best performing activation function was ... and ...

1. INTRODUCTION

Over the last few years, machine learning and neural networks have become an increasingly important part of data analysis, with a large range of applications. From image recognition to predictive analytics and scientific simulations, these techniques are reshaping the way the scientific community tackles complicated problems. As such, we wish to investigate and gain an understanding of the fundamental principles behind neural networks and how to apply them.

In this project in particular, we investigate how these techniques can be applied to the healthcare system, specifically in diagnosing malignant breast cancer tumors based on the tumor's several features. We do this by testing the performance of a neural network with logistic regression as we are working with discrete data. The various parameters are optimized using Stochastic Gradient Decent (SGD).

2. THEORY

In this section we derive and discuss the relevant theory behind the methods used.

2.1. Linear Regression

As discussed in a previous project (cite project 1), linear regression is the simplest method for fitting a continuous given a data set. The data set is approximated by

$$\boldsymbol{y} = \boldsymbol{X}\boldsymbol{\beta} \tag{1}$$

and the β coefficients are found by minimizing the cost function. For this project we consider the two regression methods

$$C_{\text{OLS}}(\boldsymbol{\beta}) = \frac{2}{n}(\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta}) \tag{2}$$

$$C_{\text{Ridge}}(\boldsymbol{\beta}) = C_{\text{OLS}}(\boldsymbol{\beta}) + \lambda||\boldsymbol{\beta}||_2^2 \tag{3}$$

in which one insists that the derivative of these w.r.t. $\boldsymbol{\beta}$ is 0 and one arrives at

$$\boldsymbol{\beta}_{\text{OLS}} = (\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T\boldsymbol{y} \tag{4}$$

$$\boldsymbol{\beta}_{\text{Ridge}} = (\boldsymbol{X}^T\boldsymbol{X} + \lambda\boldsymbol{I})^{-1}\boldsymbol{X}^T\boldsymbol{y} \tag{5}$$

which is what we will use to compute the coefficients β_i.

2.2. Logistic Regression

Whilst linear regression is quite successful in fitting continuous data such as terrain data, when the output is supposed to be discrete it fails. Linear regression predicts values across a continuous spectrum, resulting in predictions outside the range of valid class labels, such as giving negative probabilities. Logistic regression on the other hand is specifically designed for binary classification problems, and is thus ideal when dealing with discrete outcomes.

Logistic regression models the probability that a given input belongs to a particular class. It does this by applying a activation function to a linear combination of the input features. For example the sigmoid function

$$p(z) = \frac{1}{1 + e^{-z}} \tag{6}$$

maps any $z \in \mathbb{R}$ to a real number in the interval $(0, 1)$. Mathematically, logistic regression is

$$P(Y = 1|\boldsymbol{X}) = \frac{1}{1 + \exp(\boldsymbol{X}\boldsymbol{\beta})} \tag{7}$$

where $P(Y = 1|\boldsymbol{X})$ is the probability that the output Y is 1 given the input features $\boldsymbol{X}$ and coefficients $\boldsymbol{\beta}$ which describe the model. We then insert a decision boundary for classification, which in case of the sigmoid function is 1/2. This implies that if $P(Y = 1|\boldsymbol{X}) \geq 1/2$ then we say that this instance belongs to class 1, and otherwise it belongs to class 0.

The model is then trained on using the Maximum Likelihood Estimation (MLE), which finds the parameters $\boldsymbol{\beta}$ that maximize the likelihood of the observed data. The cost function is logistic regression is then the negative of

[*] e.b.rornes@fys.uio.no
[†] Insert Email

the log-MLE, i.e.

$$C(\boldsymbol{\beta}) = -\frac{1}{n}\sum_{i=1}^{n}[y_i \ln(P(Y=1|\boldsymbol{X}_i))] \tag{8}$$
$$+ (1-y_i)\ln(1-P(Y=1|\boldsymbol{X}_i)) \tag{9}$$

The particular activation functions we will use are sigmoid, tanh, ReLU and LeakyReLU shown in (10-13) respectively:

$$\sigma(z) = \frac{1}{1+e^{-z}} \tag{10}$$
$$\tanh(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}} \tag{11}$$
$$R(z) = \begin{cases} 0 & \text{if } x \leq 0 \\ z & \text{if } x > 0 \end{cases} \tag{12}$$
$$LR(z) = \begin{cases} 10^{-2}z & \text{if } x \leq 0 \\ z & \text{if } x > 0 \end{cases} \tag{13}$$

2.3. Regularization terms

2.4. Resampling Methods

2.5. Gradient Decent

Gradient decent (GD) is an essential optimization algorithm in machines learning, and is commonly used to minimize cost functions by adjusting model parameters iteratively. Given model parameters θ and a cost function $C(\theta)$ the GD algorithm is as follows:

$$\theta_i^{(j+1)} = \theta_i^{(j)} - \eta\frac{\partial C}{\partial \theta_i} \tag{14}$$

Taking it a step further we can perform batch gradient decent (BGD)

$$\theta^{(j+1)} = \theta^{(j)} - \eta\nabla_\theta C \tag{15}$$

2.6. Stochastic Gradient Decent

2.7. Neural Networks

2.7.1. Feed Forward Neural Networks

2.7.2. Back Propagation

3. IMPLEMENTATION

4. RESULTS & DISCUSSION

4.1. Linear Regression

4.2. Non-Linear Regression

4.3. Logistic Regression

5. CONCLUSION

Test bib [1]

[1] Planck: N. Aghanim *et. al.*, *Planck 2018 results. VI. Cosmological parameters*, *Astron. Astrophys.* **641** (2020) A6, [`arXiv:1807.06209`]. [Erratum: Astron.Astrophys. 652, C4 (2021)].