

Backpropagation Algorithm

Subham Roy

Department of Computer Science and Engineering
University at Buffalo, Buffalo, NY 14260

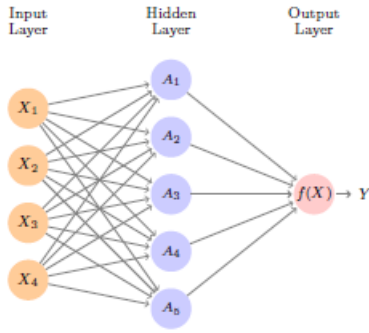
subhamro@buffalo.edu

1 Overview

A neural network is a series of algorithms that endeavors to recognize underlying relationships in a set of data through a process that mimics the way the human brain operates. In this sense, neural networks refer to systems of neurons, either organic or artificial in nature. Neural networks can adapt to changing input; so, the network generates the best possible result without needing to redesign the output criteria.

2 About Backpropagation Algorithm

Fitting neural networks are complex, and we have an overview here. The idea is to generalize much more complex networks. We start with the simple network depicted below –



In the model, the hidden layer computes activations $A_k = h_k(X)$ that are nonlinear transformations of linear combinations of the inputs X_1, X_2, \dots, X_p , and the parameters are $\beta = (\beta_0, \beta_1, \dots, \beta_K)$, as well as each of the $w_k = (w_{k0}, w_{k1}, \dots, w_{kp})$, $k = 1, \dots, K$. Given observations (x_i, y_i) , $i = 1, \dots, n$, we could fit the model by solving a nonlinear least squares problem –

$$\text{minimize}_{\{w_k\}_1^K, \beta} \frac{1}{2} \sum_{i=1}^n (y_i - f(x_i))^2,$$

where

$$f(x_i) = \beta_0 + \sum_{k=1}^K \beta_k g\left(w_{k0} + \sum_{j=1}^p w_{kj} x_{ij}\right)$$

The objective in the first formula looks simple enough, but because of the nested arrangement of the parameters and the symmetry of the hidden units, it is not straightforward to minimize. Suppose we represent all the parameters in one long vector θ . Then we can rewrite the objective in the first formula as –

$$R(\theta) = \frac{1}{2} \sum_{i=1}^n (y_i - f_\theta(x_i))^2$$

where we make explicit the dependence on the parameters.

Now our objective is to move θ to decrease the objective $R(\theta)$. The gradient of $R(\theta)$, evaluated at some latest/current value gradient $\theta = \theta^m$, is the vector of partial derivatives at that point:

$$\nabla R(\theta^m) = \left. \frac{\partial R(\theta)}{\partial \theta} \right|_{\theta = \theta^m}$$

The superscript of $\theta = \theta^m$ means that after computing the vector of derivatives, we evaluate it at the latest/current θ^m . This gives the direction in θ -space in which $R(\theta)$ increases most rapidly. The idea of gradient descent is to move θ in the opposite direction since we wish to go backward:

$$\theta^{m+1} \leftarrow \theta^m - \rho \nabla R(\theta^m)$$

For a small value of the learning rate ρ , this step will decrease the objective $R(\theta)$; i.e., $R(\theta^{m+1}) \leq R(\theta^m)$. If the gradient vector is zero, then we may have arrived at a minimum of the objective. Since, $R(\theta) = \sum_{i=1}^n R_i(\theta) = \frac{1}{2} \sum_{i=1}^n (y_i - f_\theta(x_i))^2$ is a sum, its gradient is also a sum over the n observations, so now we will just examine one of the terms –

$$R_i(\theta) = \frac{1}{2} \left(y_i - \beta_0 - \sum_{k=1}^K \beta_k g\left(w_{k0} + \sum_{j=1}^p w_{kj} x_{ij}\right) \right)^2$$

To simplify the expressions to follow, we write $z_{ik} = w_{k0} + \sum_{j=1}^p w_{kj} x_{ij}$

Firstly, we take the derivative with respect to β_k :

$$\begin{aligned}\frac{\partial R_i(\theta)}{\partial \beta_k} &= \frac{\partial R_i(\theta)}{\partial f_\theta(x_i)} \cdot \frac{\partial f_\theta(x_i)}{\partial \beta_k} \\ &= -(y_i - f_\theta(x_i)) \cdot g(z_{ik})\end{aligned}$$

And now we take the derivative with respect to w_{kj} :

$$\begin{aligned}\frac{\partial R_i(\theta)}{\partial w_{kj}} &= \frac{\partial R_i(\theta)}{\partial f_\theta(x_i)} \cdot \frac{\partial f_\theta(x_i)}{\partial g(z_{ik})} \cdot \frac{\partial g(z_{ik})}{\partial z_{ik}} \cdot \frac{\partial z_{ik}}{\partial w_{kj}} \\ &= -(y_i - f_\theta(x_i)) \cdot \beta_k \cdot g'(z_{ik}) \cdot x_{ij}.\end{aligned}$$

Note that both these expressions contain the residual $(y_i - f_\theta(x_i))$. In the result for the derivative with respect to β_k , we see that a fraction of that residual gets attributed to each of the hidden units according to the value of $g(z_{ik})$. Then in the later derivative, we see a similar attribution to input j via hidden unit k . So, the act of differentiation assigns a fraction of the residual to each of the parameters via the chain rule of differentiation, a process commonly known as Backpropagation in Neural Networks.

3 Experiment

In this experiment, I used the Women Breast Cancer dataset from the url in Reference. Initially, we start with loading the Dataset and checking the dimensions and the number of variables within the Dataset. In this case, the dataset has 32 features and 569 rows.

Dataset description –

id	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	compactness_mean	concavity_mean	concave points_mean	...
0	842302	M	17.99	10.38	122.80	1001.0	0.11840	0.27760	0.30010	0.14710
1	842517	M	20.57	17.77	132.90	1326.0	0.08474	0.07864	0.08690	0.07017
2	84300903	M	19.69	21.25	130.00	1203.0	0.10960	0.15990	0.19740	0.12790
3	84348301	M	11.42	20.38	77.58	386.1	0.14250	0.28390	0.24140	0.10520
4	84358402	M	20.29	14.34	135.10	1297.0	0.10030	0.13280	0.19800	0.10430
...
564	926424	M	21.56	22.39	142.00	1479.0	0.11100	0.11590	0.24390	0.13890
565	926682	M	20.13	28.25	131.20	1261.0	0.09780	0.10340	0.14400	0.09791
566	926954	M	16.60	28.08	108.30	858.1	0.08455	0.10230	0.09251	0.05302
567	927241	M	20.60	29.33	140.10	1265.0	0.11780	0.27700	0.35140	0.15200
568	92751	B	7.76	24.54	47.92	181.0	0.05263	0.04362	0.00000	0.00000

569 rows x 32 columns

```

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 569 entries, 0 to 568
Data columns (total 32 columns):
#   Column                               Non-Null Count  Dtype
---  ---                               -
0    id                                   569 non-null    int64
1    diagnosis                           569 non-null    object
2    radius_mean                         569 non-null    float64
3    texture_mean                       569 non-null    float64
4    perimeter_mean                     569 non-null    float64
5    area_mean                         569 non-null    float64
6    smoothness_mean                    569 non-null    float64
7    compactness_mean                   569 non-null    float64
8    concavity_mean                     569 non-null    float64
9    concave points_mean                569 non-null    float64
10   symmetry_mean                      569 non-null    float64
11   fractal_dimension_mean             569 non-null    float64
12   radius_se                          569 non-null    float64
13   texture_se                         569 non-null    float64
14   perimeter_se                       569 non-null    float64
15   area_se                           569 non-null    float64
16   smoothness_se                      569 non-null    float64
17   compactness_se                     569 non-null    float64
18   concavity_se                       569 non-null    float64
19   concave points_se                  569 non-null    float64
20   symmetry_se                        569 non-null    float64
21   fractal_dimension_se               569 non-null    float64
22   radius_worst                       569 non-null    float64
23   texture_worst                      569 non-null    float64
24   perimeter_worst                    569 non-null    float64
25   area_worst                         569 non-null    float64
26   smoothness_worst                   569 non-null    float64
27   compactness_worst                  569 non-null    float64
28   concavity_worst                     569 non-null    float64
29   concave points_worst               569 non-null    float64
30   symmetry_worst                     569 non-null    float64
31   fractal_dimension_worst            569 non-null    float64
dtypes: float64(30), int64(1), object(1)
memory usage: 142.4+ KB

```

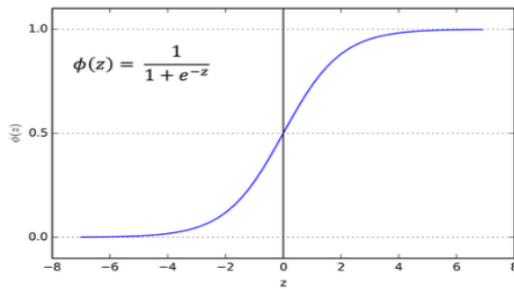


Fig: (Top Right) Dataset; Fig: (Bottom Left) Sigmoid Activation Function ; Fig: (Left) Dataset Column Datatypes

The next step is we perform data preprocessing. We perform the following steps before we can be certain that we can apply the classifier model. This is almost like all the techniques we apply. First, we check for missing data, then check for the spread of the data like where the Mean, Median of the data, are there any outliers, and we take care of that.

In this dataset, there were null values. After that, we label encoded the Diagnosis column to 0 and 1 i.e., whether the patient does not have cancer or do they. After that, I scaled the Train and Test data separately to remove any chance of complications using StandardScaler. The Train and Test split was 80:20.

Below are the X-train set and Y-train set from Left to Right –

```
array([[ -0.52787029,  2.49821982, -0.59939466, ..., -1.74713139,
        -0.79044533, -0.91054389],
       [ -0.55333608,  0.29431013, -0.60759343, ..., -0.62275667,
        -0.33646358, -0.83551633],
       [ 2.15452653,  0.40392257,  2.26525805, ...,  1.03846122,
        -0.11504791,  0.26488788],
       ...,
       [-1.3297598 , -0.21876938, -1.32088704, ..., -0.98271999,
        -0.718764  , -0.13637062],
       [-1.24940108, -0.24209117, -1.2835826 , ..., -1.74713139,
        -1.58690456, -1.01280367],
       [-0.74291476,  1.08958336, -0.71827692, ..., -0.2865488 ,
        -1.26354211,  0.19486216]])
```

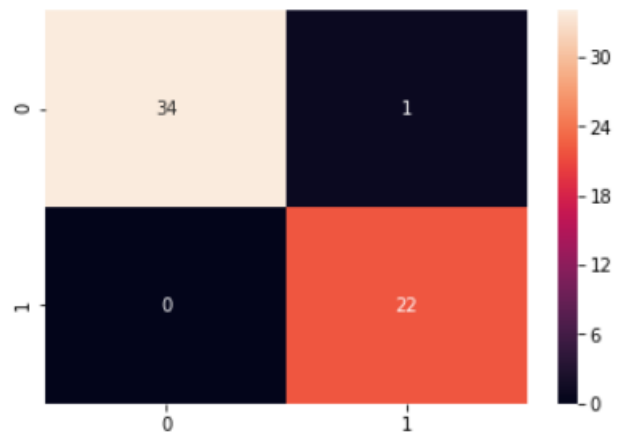
Fig: X-train set, and Y-train set (from Left to Right)

Now we implement the Neural network model on the Training set.

4 Analysis

We find out firstly, the model accuracy for on Test dataset is 98%, which is good considering the data was so complicated. The below confusion matrix depicts that for the most part the prediction for Diagnosis i.e., (M = malignant, B = benign) is done correctly.

Except for one case where the model predicted, a False negative, which means the patient had cancer, but the model predicted they did not. Since the overall prediction is extremely accurate, we can work with this model in the future for new test data.



References

- [1] Gareth James [at], Daniela Witten [aut], Trevor Hastie [aut, cre], Rob Tibshirani [aut], Balasubramanian Narasimhan [ctb], *Introduction to Statistical Learning, Second Edition*.
- [2] Tom M. Mitchell, *Machine Learning: A multistrategy approach, 1997 Edition*.
- [3] Breast Cancer Data: URL - <https://archive.ics.uci.edu/ml/datasets/Breast+Cancer+Wisconsin+%28Diagnostic%29>