Introduction to neural networks

Feed-forward networks

Juan Irving Vásquez
jivg.org

Centro de Innovación y Desarrollo Tecnológico en Cómputo
Instituto Politécnico Nacional

September 11, 2023©

Contents

Roadmap

1. Simplest neural network
2. Gradient descent
3. Networks with multiple outputs
4. Practical aspects

The Simplest Neural Network (NN)

One layer network

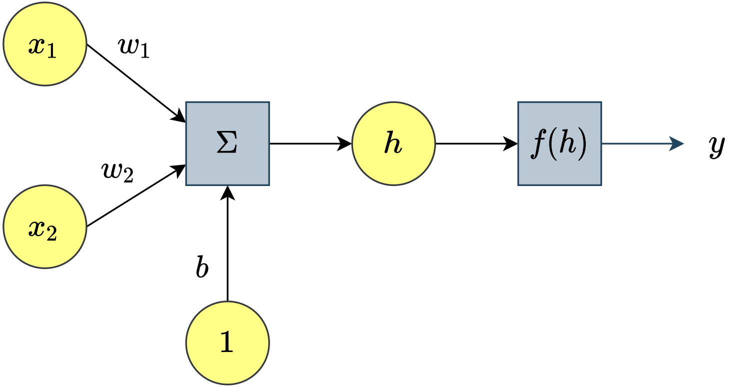
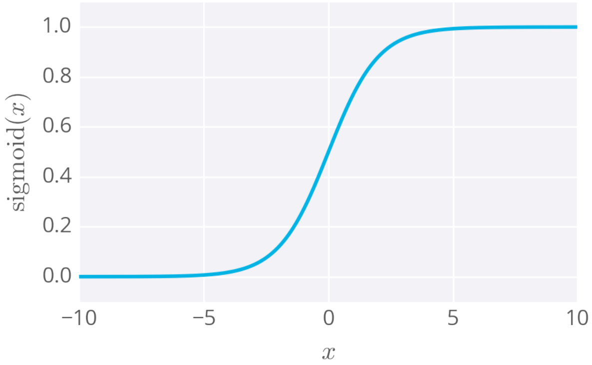


Figure: Diagram of a simple neural network (one perceptrón)

- The activation function can be any function.
- A nice function is the sigmoid (logistic)

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



The simpest NN

$$\hat{y} = f(h) = sigmoid(\sum_{i=1}^{n} w_i x_i + b) \tag{2}$$

- A simple NN does not offer advantage over linear regression models.
- Stacking nodes provides a powerful tool versus regression models.

Figure: Ilustración de aprendizaje. [1]

[1] Anderson-bastidas.com

Prediction Error

- How wrong the predictions are?
- Sum of squared errors (SSE)[2]

$$E = \frac{1}{2} \sum_{\mu=1}^{m} [y^{\mu} - \hat{y}^{\mu}]^2 \tag{3}$$

where $\hat{y}$ is the prediction, y is the true value and μ the data row.

[2] *divided by 2 for further derivation

Learning the weights

- Prediction

$$\hat{y}^{\mu} = f\left(\sum_{i=1}^{n} w_i x_i^{\mu} + b\right) \tag{4}$$

- The error depends on the weights

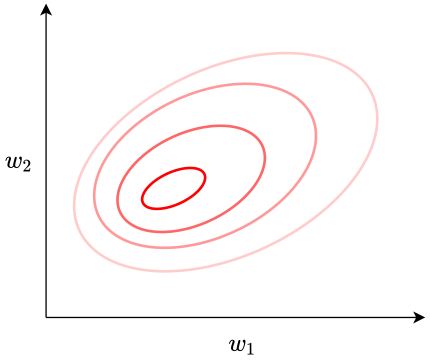
$$E(w_i) = \frac{1}{2}\sum_{\mu=1}^{m}\left[y^{\mu} - f\left(\sum_{i=1}^{n} w_i x_i^{\mu} + b\right)\right]^2 \tag{5}$$

- Our goal is to find the weights that minimice the error.

Roadmap

Gradient Descent

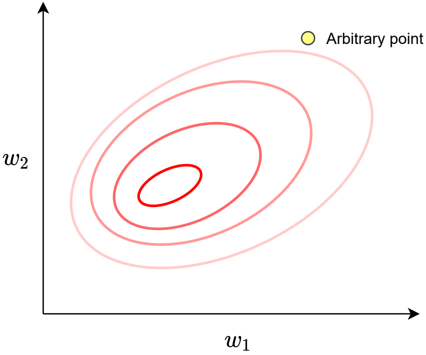
- At each step the error and the gradient are calculated, then the weights are calculated [3].



[3] Rosenbloom, P. "The method of steepest descent." Proc Symp Appl Math. Vol. 6. 1956.

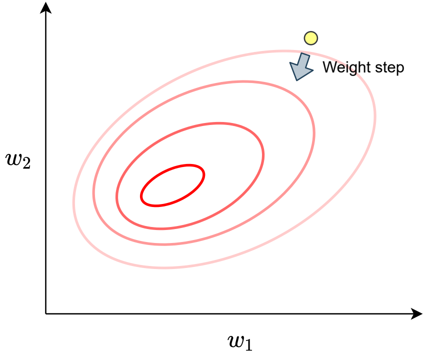
Gradient Descent

- At each step the error and the gradient are calculated, then the weights are calculated.



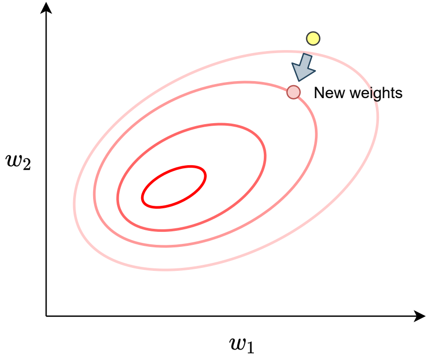
Gradient Descent

- At each step the error and the gradient are calculated, then the weights are calculated.



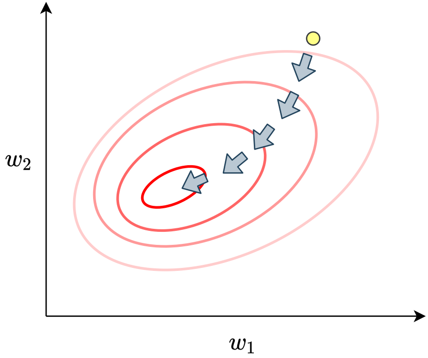
Gradient Descent

- At each step the error and the gradient are calculated, then the weights are calculated.



Gradient Descent

- At each step the error and the gradient are calculated, then the weights are calculated.



- If f is a function of two variables x and y, then the gradient of f is the vector function ∇f defined by

$$\nabla f(x,y) = [f_x(x,y), f_y(x,y)] = \frac{\partial f}{\partial x} i + \frac{\partial f}{\partial y} j \tag{6}$$

[Stewart, Multivariate Calculus]

Gradient

- It is a vector valued function.

$$y = f(w_1, w_2, \times, w_n)$$

$$\nabla f = \begin{bmatrix} \frac{\partial f}{\partial w_1} \\ \frac{\partial f}{\partial w_2} \\ \dots \\ \frac{\partial f}{\partial w_n} \end{bmatrix}$$

Gradient

- It is a vector valued function.

$$y = f(w_1, w_2, \times, w_n)$$

$$\nabla f = \begin{bmatrix} \frac{\partial f}{\partial w_1} \\ \frac{\partial f}{\partial w_2} \\ \cdots \\ \frac{\partial f}{\partial w_n} \end{bmatrix}$$

- Example

$$f(x, y) = \begin{bmatrix} xy \\ xy \end{bmatrix}$$

$$\nabla f = ?$$

Gradient

- It is a vector valued function.

$$y = f(w_1, w_2, \times, w_n)$$

$$\nabla f = \begin{bmatrix} \frac{\partial f}{\partial w_1} \\ \frac{\partial f}{\partial w_2} \\ \dots \\ \frac{\partial f}{\partial w_n} \end{bmatrix}$$

- Example

$$f(x, y) = \begin{bmatrix} xy \\ xy \end{bmatrix}$$

$$\nabla f = ?$$

$$\nabla f = \begin{bmatrix} y \\ x \end{bmatrix}$$

- A nice explanation: khanacademy.org/.../gradient

Gradient descent

- The gradient of the error function

$$\frac{\partial E}{\partial w_i} = -(y - \hat{y})f'(h)x_i \tag{7}$$

- Error term

$$\delta = (y - \hat{y})f'(h) = (y - \hat{y})f'(\sum_i w_i x_i) \tag{8}$$

- Weight step

$$\Delta w_i = \eta \delta x_i \tag{9}$$

- Weight update

$$w_i = w_i + \Delta w_i \tag{10}$$

Data: Conjunto de datos (*Data*), Número de registros (m)
Result: Pesos entrenados (W)

```
Δw_i = 0 ;
w_i = Initialize();                                  /* Inicializar pesos */
for e epocas do
    foreach x ∈ Data do
        ŷ = f(h(x, W)) ;                             /* Pase frontal */
        δ = (y − ŷ)f′(h) ;                           /* Término de error */
        Δw_i = Δw_i + δx_i ;                         /* Acumular Incremento */
    end
    w_i = w_i + (η/m)Δw_i ;                          /* Actualizar pesos */
end
return W
```

$\Delta w_i = 0$;
$w_i =$ Initialize(); /* Inicializar pesos */
for *e epocas* **do**
  foreach $x \in Data$ **do**
    $\hat{y} = f(h(x, W))$; /* Pase frontal */
    $\delta = (y - \hat{y})f'(h)$; /* Término de error */
    $\Delta w_i = \Delta w_i + \delta x_i$; /* Acumular Incremento */
  end
  $w_i = w_i + \frac{\eta}{m}\Delta w_i$; /* Actualizar pesos */
end
return W

Algorithm 1: Descenso por gradiente

Exercise. Implement a simple neural network.

- Given

$$x = [2, 1],$$
$$w = [-0.5, 0.5],$$
$$y = 0.6,$$
$$\eta = 0.4,$$
$$f = sigmoid$$

- Draw the diagram of the NN.
- Calculate: i) output of the NN, ii) residual error of the NN, iii) error term and iv) weight increment,

- Derivative of the Sigmoid

$$f(x) = sigmoid(x) = \frac{1}{1+e^{-x}} \tag{11}$$

$$f' = f\,(1-f) \tag{12}$$

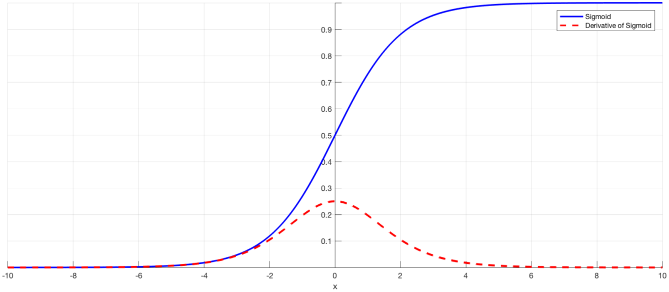


Figure: Sigmoid and its derivative

Roadmap

Multiple outputs

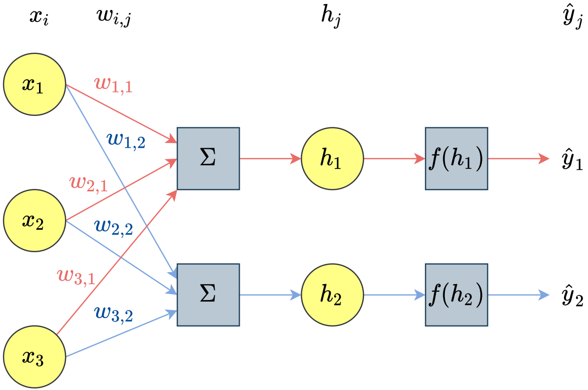


Figure: Neural network with multiple outputs

- Prediction

$$\hat{y}_j^{\mu} = f\left(\sum_i w_{i,j} x_i + b\right) \tag{13}$$

- Sum of squared errors (SSE)

$$E = \frac{1}{2} \sum_{\mu} \sum_{j} \left[y_j^{\mu} - \hat{y}_j^{\mu} \right]^2 \tag{14}$$

where $\hat{y}$ is the prediction, y is the true value, j the output and μ the data row. Substituting:

$$E = \frac{1}{2} \sum_{\mu} \sum_{j} \left[y_j^{\mu} - f\left(\sum_i w_{i,j} x_i + b\right) \right]^2 \tag{15}$$

Learning the weights for multiple outputs

- Error term

$$\delta_j = (y_j - \hat{y}_j) f'(h_j) \tag{16}$$

- Weight step

$$\Delta w_{i,j} = \eta \delta_j x_i \tag{17}$$

- Weight update

$$w_{i,j} = w_{i,j} + \Delta w_{i,j} \tag{18}$$

Roadmap

Mean squared error

- Avoids the divergence from the gradient descent provoked by a large data set.

$$E = \frac{1}{2m} \sum_{\mu} (y^{\mu} - \hat{y}^{\mu})^2 \tag{19}$$

- m is the number of examples

Mean squared error

Loss functions

- Smaller penalization to large errors

$$E = \frac{1}{2m} \sum_{\mu} |y^{\mu} - \hat{y}^{\mu}| \tag{20}$$

- m is the number of examples

Weights initialization

- Set bias (b) to zero
- Zero initialization for weights is not successful
- Random numbers is better but be careful of large numbers

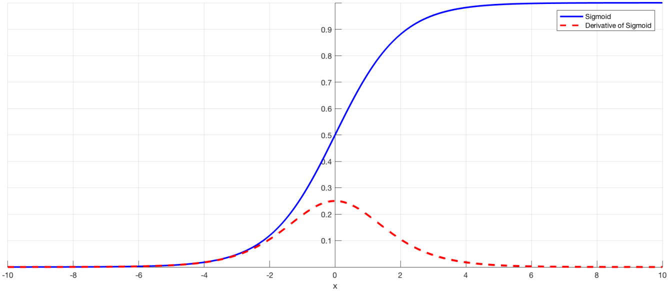
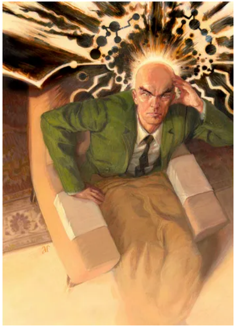


Figure: Sigmoid and its derivative

Weights initialization (2)



- Xavier initialization.

$$w = rand(-\frac{1}{\sqrt{n}}, \frac{1}{\sqrt{n}}) \tag{21}$$

where n is number of input units.

Categorical Data

- Admissions (again..). This dataset has three input features: GRE score, GPA, and the rank of the undergraduate school (numbered 1 through 4). Institutions with rank 1 have the highest prestige, those with rank 4 have the lowest.

Categorical Data

One-hot encoding

- Numerical and categorical data
- Add dummy variables.

	admit	gre	gpa	rank_1	rank_2	rank_3	rank_4
15	0	-0.932334	0.131646	0	0	1	0
115	0	0.279614	1.576859	0	0	1	0
55	1	1.318426	1.603135	0	0	1	0
175	1	0.279614	-0.052290	0	1	0	0
63	1	0.799020	1.208986	0	0	1	0
67	0	0.279614	-0.236227	1	0	0	0
216	0	-2.144282	-1.287291	1	0	0	0
145	0	-1.798011	0.105369	0	0	1	0
286	1	1.837832	-0.446439	1	0	0	0
339	1	0.625884	0.210476	0	0	1	0

References

- Goodfellow, Ian, et al. Deep learning. Vol. 1. No. 2. Cambridge: MIT press, 2016.
- Stewart, James, Daniel K. Clegg, and Saleem Watson. Calculus: early transcendentals. Cengage Learning, 2020.
- Udacity - Self Driving Car Nanodegree
- Khan Academy