Machine Learnia Pseudo-Code Blobs Perceptron Cost Gradient Vector Algorithm

Spike Spiegel

December 26, 2025

Contents

1 Pseudo-code

1.1 Matrix transpose

```
Function transpose_matrix(M):
    m = number of rows in M
    n = number of columns in M
    T = empty matrix

    For j from 0 to n-1:
        Initialize row as an empty list
        For i from 0 to m-1:
            Append M[i][j] to row
        Append row to T

    Return T
```

1.2 Matrix Multiplication

```
Function multiply-matrices(matrix1, matrix2):
    m = number of rows in matrix1
    n = number of columns in matrix1
    p = number of columns in matrix2

    If n  number of rows in matrix2:
        Return "Error: incompatible dimensions"

    Initialize result as an empty list

    For i from 0 to m-1:
        Initialize row as an empty list
        For j from 0 to p-1:
            sum = 0
            For k from 0 to n-1:
                sum = sum + matrix1[i][k] * matrix2[k][j]
            Append sum to row
        Append row to result

    Return result
```

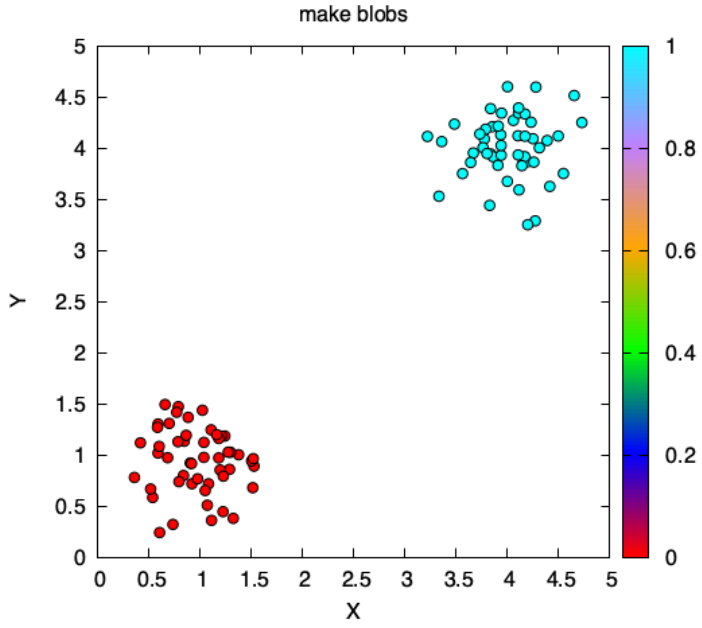
2 Blobs

2.1 Objective

Build a data set with:

- X: 2D points (or N dimensions)
- y: cluster label

Each cluster is centralize around one center with gaussian noise. Just a simple example



3 Perceptron

3.1 Linear function

$$z = w_1 x_1 + w_2 x_2 + b$$

3.2 Sigmoid function

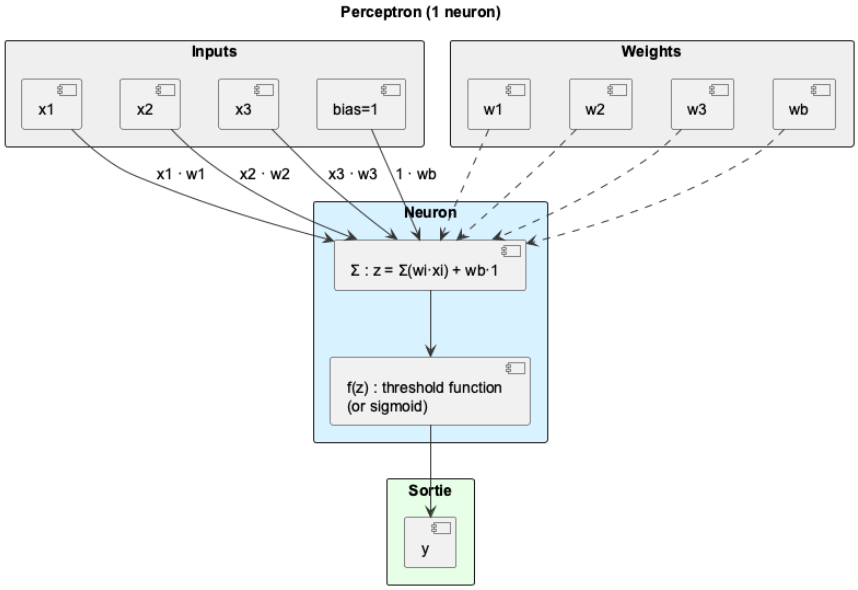
$$a(z) = \frac{1}{1 + e^{-z}}$$

3.3 Bernoulli

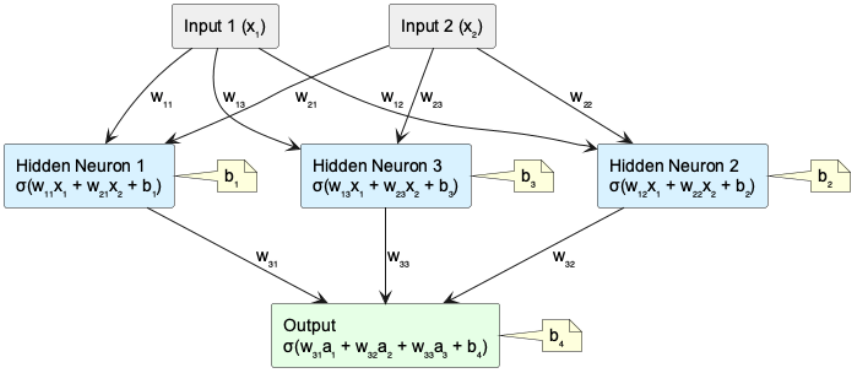
$$P(Y = y) = a(z)^y \times (1 - a(z))^{1-y}$$
$$P(Y = 0) = \times(1 - a(z))$$
$$P(Y = 1) = a(z)$$

4 Graphic

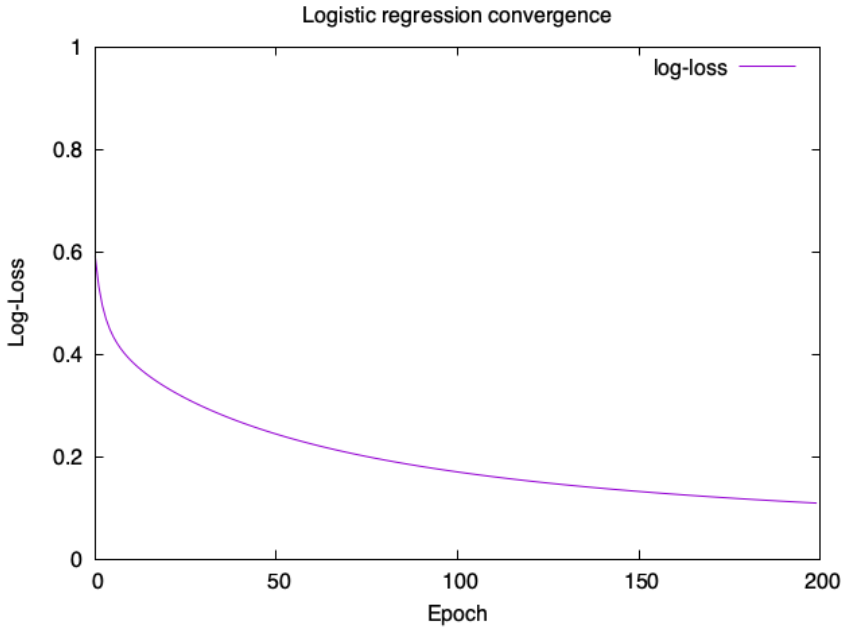
4.1 Overview



4.2 Neuron Network



4.3 Log Loss Convergence



5 Cost function

5.1 Likelihood (*Vraisemblance*)

$$\begin{aligned} L &= \prod_{i=1}^{m} P(Y = y_i) \\ &= \prod_{i=1}^{m} a_i^{y_i} \times (1 - a_i)^{1-y_i} \end{aligned}$$

5.2 Loss function (Log Loss)

$$\mathcal{L} = -\frac{1}{m} \sum_{i=1}^{m} (y_i log(a_i) + (1 - y) log(1 - a_i))$$

5.3 Log Likelihood

$$\begin{aligned} LL &= log(\prod_{i=1}^{m}(a_i^{y_i} \times (1-a_i)^{1-y_i}) \\ &= \sum_{i=1}^{m}(log(a_i^{y_i}) + log(1-a_i)^{1-y_i}) \\ &= \sum_{i=1}^{m}(y_i log(a_i) + (1-y_i) log(1-a_i)) \end{aligned}$$

5.4 Result

$$\mathcal{L} = -\frac{1}{m} \times (LL)$$

6 Gradient descent

This involves adjusting the parameters **W** and **b** in order to minimize model errors, i.e., to minimize the **cost function** (Log Loss).
That is why we calculate the gradient (or derivative) of the **cost function**.

6.1 Gradient descent algorithm

$$W = W - \alpha \times \frac{\partial \mathcal{L}}{\delta W}$$
$$b = b - \alpha \times \frac{\delta \mathcal{L}}{\delta b}$$

6.2 Decomposition

$$\frac{\delta \mathcal{L}}{\delta w_1} = \frac{\delta \mathcal{L}}{\delta a} \times \frac{\delta a}{\delta z} \times \frac{\delta z}{\delta w_1}$$

6.3 Calculation of $\frac{\delta \mathcal{L}}{\delta a}$

$$\begin{aligned}\frac{\delta \mathcal{L}}{\delta a} &= -\frac{1}{m}\sum_{i=1}^{m}(\frac{y_i}{a_i} - \frac{1-y_i}{1-a_i}) \\ &= -\frac{1}{m}\sum_{i=1}^{m}(\frac{y_i - a_i y_i - a_i + a_i y_i}{a_i(1-a_i)}) \\ &= -\frac{1}{m}\sum_{i=1}^{m}(\frac{y_i - a_i}{a_i(1-a_i)})\end{aligned}$$

6.4 Calculation of $\frac{\delta a}{\delta z}$

Let us choose $h = g \circ f$ with $f(z) = 1 + e^{-z}$ and $g(f(z)) = \frac{1}{f(z)}$, i.e.

$$\begin{aligned}h' &= g'(f(z))f'(z) \\ &= -\frac{1}{f(z)^2} \times (-e^{-z}) \\ &= \frac{e^{-z}}{(1+e^{-z})^2} \\ &= \frac{1-1+e^{-z}}{(1+e^{-z})^2} \\ &= -\frac{1}{(1+e^{-z})^2} + \frac{1+e^{-z}}{(1+e^{-z})^2} \\ &= \frac{1}{1+e^{-z}} - \frac{1}{(1+e^{-z})^2} \\ &= (\frac{1}{1+e^{-z}})(1 - \frac{1}{1+e^{-z}})\end{aligned}$$

In conclusion, we obtain

$$\frac{\delta a}{\delta z} = a(z)(1-a(z))$$

6.5 Calculation of $\frac{\delta z}{\delta w_1}$

$$\begin{aligned}z &= w_1 x_1 + w_2 x_2 + b \\ \frac{\delta z}{\delta w_1} &= x_1\end{aligned}$$

6.6 Calculation of $\frac{\delta z}{\delta w_2}$

$$z = w_1 x_1 + w_2 x_2 + b$$
$$\frac{\delta z}{\delta w_2} = x_2$$

6.7 Calculation of $\frac{\delta z}{\delta b}$

$$z = w_1 x_1 + w_2 x_2 + b$$
$$\frac{\delta z}{\delta b} = 1$$

6.8 Conclusion for $\frac{\delta \mathcal{L}}{\delta w_1}$

$$\begin{aligned}\frac{\delta \mathcal{L}}{\delta w_1} &= \frac{\delta \mathcal{L}}{\delta a} \times \frac{\delta a}{\delta z} \times \frac{\delta z}{\delta w_1} \\ &= (-\frac{1}{m}\sum_{i=1}^{m}(\frac{y_i - a_i}{a_i(1-a_i)}))(a(z)(1-a(z)))(x_1) \\ &= -\frac{1}{m}\sum_{i=1}^{m}(y_i - a_i)x_1\end{aligned}$$

6.9 Conclusion for $\frac{\delta \mathcal{L}}{\delta w_2}$

$$\begin{aligned}\frac{\delta \mathcal{L}}{\delta w_2} &= \frac{\delta \mathcal{L}}{\delta a} \times \frac{\delta a}{\delta z} \times \frac{\delta z}{\delta w_2} \\ &= (-\frac{1}{m}\sum_{i=1}^{m}(\frac{y_i - a_i}{a_i(1-a_i)}))(a(z)(1-a(z)))(x_2) \\ &= -\frac{1}{m}\sum_{i=1}^{m}(y_i - a_i)x_2\end{aligned}$$

6.10 Conclusion for $\frac{\delta \mathcal{L}}{\delta b}$

$$\begin{aligned}\frac{\delta \mathcal{L}}{\delta b} &= \frac{\delta \mathcal{L}}{\delta a} \times \frac{\delta a}{\delta z} \times \frac{\delta z}{\delta b} \\ &= (-\frac{1}{m}\sum_{i=1}^{m}(\frac{y_i - a_i}{a_i(1-a_i)}))(a(z)(1-a(z))) \\ &= -\frac{1}{m}\sum_{i=1}^{m}(y_i - a_i)\end{aligned}$$

6.11 Conclusion

For our gradient descent

$$W = W - \alpha \frac{\partial \mathcal{L}}{\delta W}$$
$$b = b - \alpha \frac{\delta \mathcal{L}}{\delta b}$$

we obtain the following gradients:

$$\frac{\delta \mathcal{L}}{\delta w_1} = -\frac{1}{m}\sum_{i=1}^{m}(y_i - a_i)x_1$$
$$\frac{\delta \mathcal{L}}{\delta w_2} = -\frac{1}{m}\sum_{i=1}^{m}(y_i - a_i)x_2$$
$$\frac{\delta \mathcal{L}}{\delta b} = -\frac{1}{m}\sum_{i=1}^{m}(y_i - a_i)$$

That mean:

$$w_1 = w_1 - \alpha \times \frac{1}{m}\sum_{i=1}^{m}(y_i - a_i)x_1$$
$$w_2 = w_2 - \alpha \times \frac{1}{m}\sum_{i=1}^{m}(y_i - a_i)x_2$$
$$b = b - \alpha \times \frac{1}{m}\sum_{i=1}^{m}(y_i - a_i)$$

7 Vector

7.1 Introduction

In mathematics and programming, a column vector is usually represented as a vertical list of elements, enclosed in square brackets or parentheses. In Lisp (and more specifically in Common Lisp), there is no native data type for column vectors as in mathematics or NumPy. However, you can represent a column vector as a simple list, where each element corresponds to a component of the vector.

7.2 Dataset

Let's take the case of a dataset containing m data points, each of which consists of n parameters (variables).

$$X = \begin{bmatrix} x_1^1 & \dots & x_n^1 \\ x_1^2 & \dots & x_n^2 \\ \vdots & \dots & \vdots \\ x_1^m & \dots & x_n^m \end{bmatrix} \in \mathbb{R}^{m \times n}$$

and

$$Y = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_m \end{bmatrix} \in \mathbb{R}^{m \times 1}$$

as well as

$$Z = \begin{bmatrix} z^1 \\ z^2 \\ \vdots \\ z^n \end{bmatrix}$$

For $n = 2$, we therefore find that

$$Z = \begin{bmatrix} z^1 \\ z^2 \\ \vdots \\ z^n \end{bmatrix} = \begin{bmatrix} w_1 x_1^1 + w_2 x_2^1 + b \\ w_1 x_1^2 + w_2 x_2^2 + b \\ \vdots \\ w_1 x_1^1 + w_2 x_2^1 + b \end{bmatrix} = \underbrace{\begin{bmatrix} x_1^1 & x_2^1 \\ x_1^2 & x_2^2 \\ \vdots & \vdots \\ x_1^m & x_2^m \end{bmatrix}}_{(m,2)} \cdot \underbrace{\begin{bmatrix} w_1 \\ w_2 \end{bmatrix}}_{(2,1)} + \underbrace{\begin{bmatrix} b \\ b \\ \vdots \\ b \end{bmatrix}}_{(m,1)}$$

$$Z = X.W + b$$

or

$$Z = \begin{bmatrix} z^1 \\ z^2 \\ \vdots \\ z^n \end{bmatrix} = \begin{bmatrix} w_1 x_1^1 + w_2 x_2^1 + b \\ w_1 x_1^2 + w_2 x_2^2 + b \\ \vdots \\ w_1 x_1^1 + w_2 x_2^1 + b \end{bmatrix} = \underbrace{\begin{bmatrix} w_1 & w_2 \end{bmatrix}}_{(1,2)} \cdot \underbrace{\begin{bmatrix} x_1^1 & x_1^2 & \dots & x_1^m \\ x_2^1 & x_2^2 & \dots & x_2^m \end{bmatrix}}_{(2,m)}$$

$$Z = W^T X + b$$

7.3 Vectorization of A

Reminder

$$a^i = \sigma(z^i) = \frac{1}{1 + e^{-z^i}}$$

So, we have

$$A = \begin{bmatrix} a^1 \\ a^2 \\ \vdots \\ a^m \end{bmatrix} = \sigma \left(\begin{bmatrix} z^1 \\ z^2 \\ \vdots \\ z^m \end{bmatrix} \right) = \sigma(Z)$$

7.4 Cost function Vectorization

The objective is to compare vector A with vector y.

$$\mathcal{L} = -\frac{1}{m} \sum_{i=1}^{m} y_i log(a_i) + (1 - y_i) log(1 - a_i)$$

$$\mathcal{L} = -\frac{1}{m} \sum_{i=1}^{m} y log(A) + (1 - y) log(1 - A)$$

7.5 Gradient descent Vectorization

Remind

$$w_1 = w_1 - \alpha \frac{\delta \mathcal{L}}{\delta w_1}$$

$$w_2 = w_2 - \alpha \frac{\delta \mathcal{L}}{\delta w_2}$$

$$b = b - \alpha \frac{\delta \mathcal{L}}{\delta b}$$

So, we can write

$$\begin{bmatrix} w_1 \\ w_2 \end{bmatrix} = \begin{bmatrix} w_1 - \alpha \frac{\delta \mathcal{L}}{\delta w_1} \\ w_2 - \alpha \frac{\delta \mathcal{L}}{\delta w_2} \end{bmatrix} = \begin{bmatrix} w_1 \\ w_2 \end{bmatrix} - \alpha \begin{bmatrix} \frac{\delta \mathcal{L}}{\delta w_1} \\ \frac{\delta \mathcal{L}}{\delta w_2} \end{bmatrix} = W - \alpha \frac{\delta \mathcal{L}}{\delta W}$$

$$W = W - \alpha \frac{\delta \mathcal{L}}{\delta W}$$

$$b = b - \alpha \frac{\delta \mathcal{L}}{\delta b}$$

Mathematically, we should rather write $W_{t+1} = W_t - \alpha \frac{\delta \mathcal{L}}{\delta W}$

7.6 Gradient Vectorization

$$\frac{\delta \mathcal{L}}{\delta W} = \begin{bmatrix} \frac{\delta \mathcal{L}}{\delta w_2} \\ \frac{\delta \mathcal{L}}{\delta w_2} \end{bmatrix} = \begin{bmatrix} -\frac{1}{m} \sum_{i=1}^{m} (y_i - a_i) x_1^i \\ -\frac{1}{m} \sum_{i=1}^{m} (y_i - a_i) x_2^i \end{bmatrix} = -\frac{1}{m} \begin{bmatrix} \sum_{i=1}^{m} (y_i - a_i) x_1^i \\ \sum_{i=1}^{m} (y_i - a_i) x_2^i \end{bmatrix}$$

$$\frac{\delta \mathcal{L}}{\delta W} = -\frac{1}{m} \underbrace{\begin{bmatrix} x_1^1 & x_1^2 & \dots & x_1^m \\ x_2^1 & x_2^2 & \dots & x_2^m \end{bmatrix}}_{X^T} \cdot \left(\underbrace{\begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_m \end{bmatrix}}_{y} - \underbrace{\begin{bmatrix} a_1 \\ a_2 \\ \vdots \\ a_m \end{bmatrix}}_{A} \right)$$

$$\frac{\delta \mathcal{L}}{\delta W} = -\frac{1}{m} X^T . (y - A)$$

and

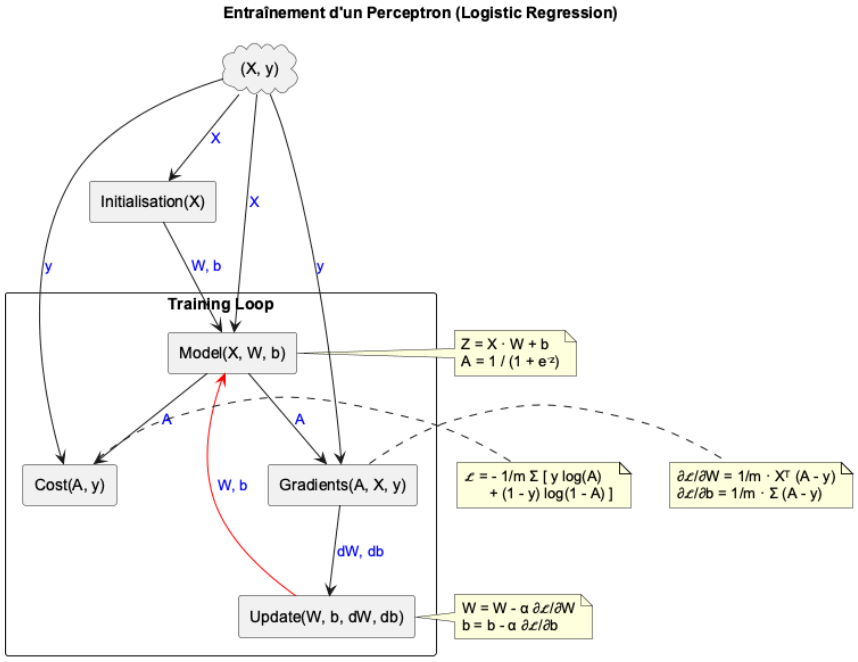
$$\frac{\delta \mathcal{L}}{\delta b} = -\frac{1}{m} \sum_{i=1}^{m} (y^i - a^i)$$

$$\frac{\delta \mathcal{L}}{\delta b} = -\frac{1}{m} \sum_{i=1}^{m} \left(\begin{bmatrix} y^1 \\ y^2 \\ \vdots \\ y^m \end{bmatrix} - \begin{bmatrix} a^1 \\ a^2 \\ \vdots \\ a^m \end{bmatrix} \right)$$

$$\frac{\delta \mathcal{L}}{\delta b} = -\frac{1}{m} \sum_{i=1}^{m} (y - A)$$

8 Algorithm

8.1 Overview



8.2 Algorithme