Neural Networks

In this class, NN is approached from a slightly different perspective, which is an extension of logistic regression.

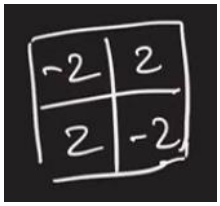
Example:

Given a 2-by-2 grid G similar to:

Find a function f that can identify this exact diagonal pattern.

For Logistic regression, we assign weights to each cell of the grid, and calculate the weighted sum. If the weighted sum reaches some threshold b, then the grid is a diagonal; otherwise, it is not.

$$w_1 a_{00} + w_2 a_{01} + w_3 a_{10} + w_4 a_{11} = \begin{cases} \geq b \\ < b \end{cases}$$

E.g., The weights are: 

; apply these weights:

$$-2 * -1 + 2*1 + 2*1 -2 * -1 = 8$$

But the threshold b may not be a fixed number; it may have many possibilities. Instead of manually setting the threshold, we move it to the left of the equation. Therefore, the weighted sum becomes:

$$w_1 a_{00} + w_2 a_{01} + w_3 a_{10} + w_4 a_{11} + b = \begin{cases} \geq 0 \\ < 0 \end{cases}$$

Moreover, instead of only having 2 colors C_1 and C_2, we have a continuum of colors $[C_1, C_2]$. Flatten the grid G to a matrix A:

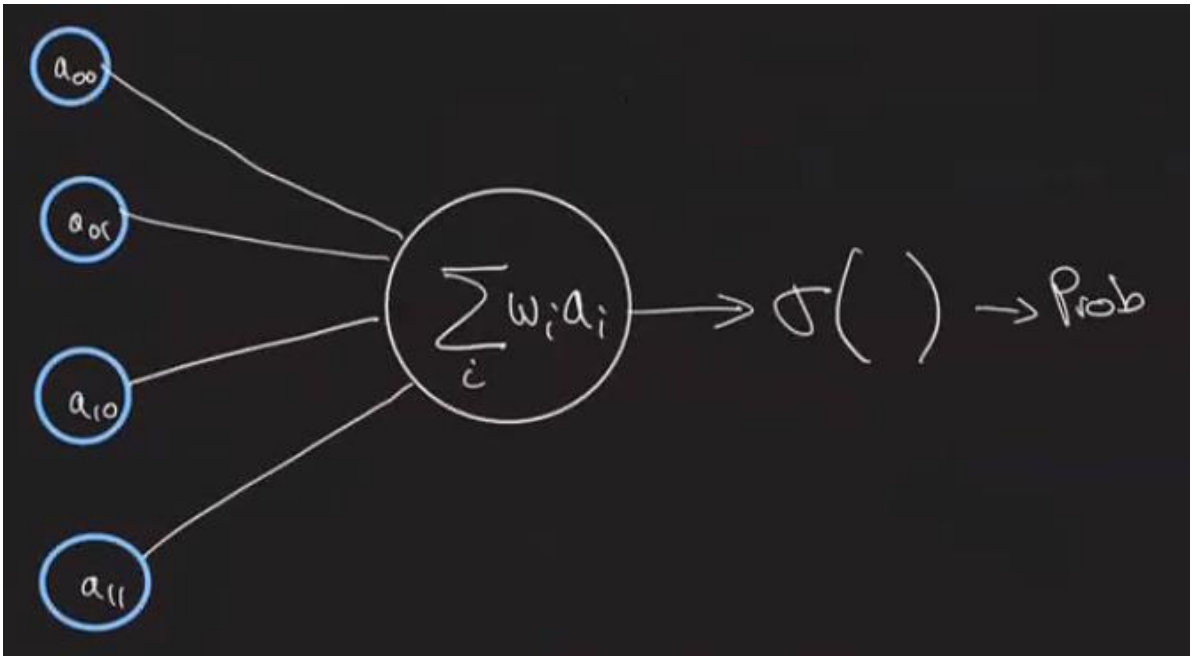
$[a_{00}\ a_{01}\ a_{10}\ a_{11}]^T$, where each a_{ij} is a color

f(A) -> 0 or 1

In the form of logistic regression:

$$f(A) = \frac{1}{1+e^{-w\,a+b}}$$

which is a sigmoid function (ϭ), where w is the weight matrix [w_1 w_2 w_3 w_4], a is the matrix A above, and b is the bias term that accounts for the systemic dimming or brightening of the grid.



Steps shown in the graph: Sum up the weights, applied sigmoid function, get the probability.

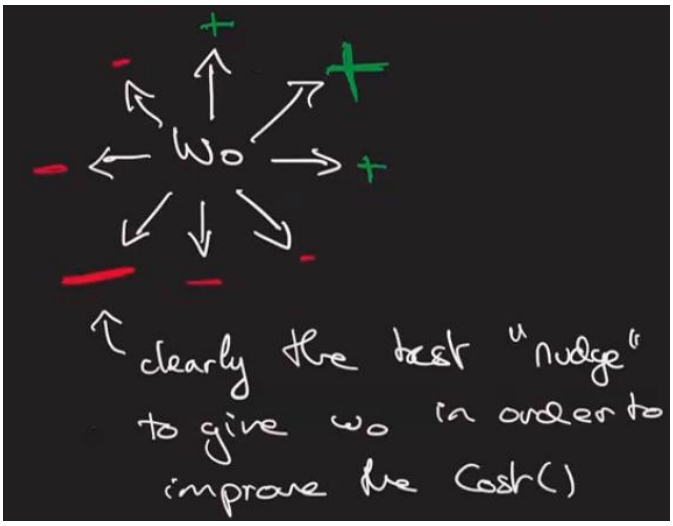
Recall for logistic regression, the goal is to:

$$\text{Max} \prod_{i=1}^{n} P(Y_i = 1 \mid X_i)$$

Which is equal to min Cost(w, b):

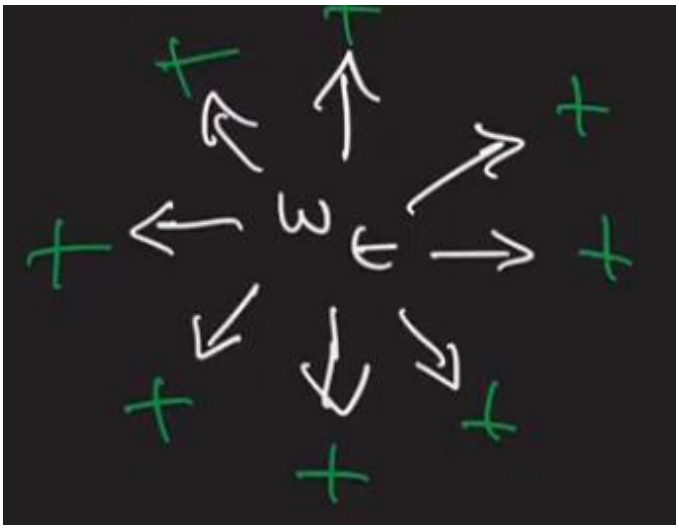
$$Min\ -\frac{1}{n}\sum_{i=1}^{n}[\,Y_i \log\left(\frac{1}{1+e^{-w\,X_i+b}}\right) + \ (1-Y_i)\log\left(1-\frac{1}{1+e^{-w\,X_i+b}}\right)\,]$$

Suppose we start from w_0 = random weight, assume there is no bias term. Look at the cost around w_0:



then, we move w_0 toward the "nudge" to w_1

By repeating the process, we would like to reach a weight w_t that looks like



Note that w_t might be a local minimum

How do we know what the direction of the best "nudge" is?

The direction of which the cost decreases most rapidly / the downward rate of change is highest.

Gradients

Best nudge should be in the direction of the gradient, which is the steepest rate of change (in a negative sense).

e.g., $\nabla f(x,y,z) = \frac{\partial f}{\partial x}\,\boldsymbol{i} + \frac{\partial f}{\partial y}\,\boldsymbol{j} + \frac{\partial f}{\partial z}\,\boldsymbol{k}$, where ***i, j, k*** are unit vectors in directions of x, y, z

e.g., $f(x,y) = 1.5\,x^2 - 2\,y$, $\nabla f = (3x)\,\boldsymbol{i} + (-2)\,\boldsymbol{j}$

At point P (0,0), $\nabla f = 3*0\,\boldsymbol{i} - 2\,\boldsymbol{j} = -2\,\boldsymbol{j}$.

If we move -2 units in the direction of ***j*** and if ***j*** = [0, 1], then

$$P' = \alpha \nabla f_{(0,0)} + P$$

$$= \alpha\,[\,0, -2\,] + [\,0, 0\,]$$

$$= [\,0, -2\alpha\,]$$

Gradient Descent

∇f : direction of steepest **increase**

$-\nabla f$: direction of steepest **decrease** (we want this to minimize the cost!)

Steps:

1. Define α in a way that is neither too big or small
2. Initialize P to be random
3. $P_{new} = -\,\alpha \nabla f_p + P$
4. $P <= P_{new}$
5. Repeat step 3 and 4 until $P \approx P_{new}$ (the change is very small)