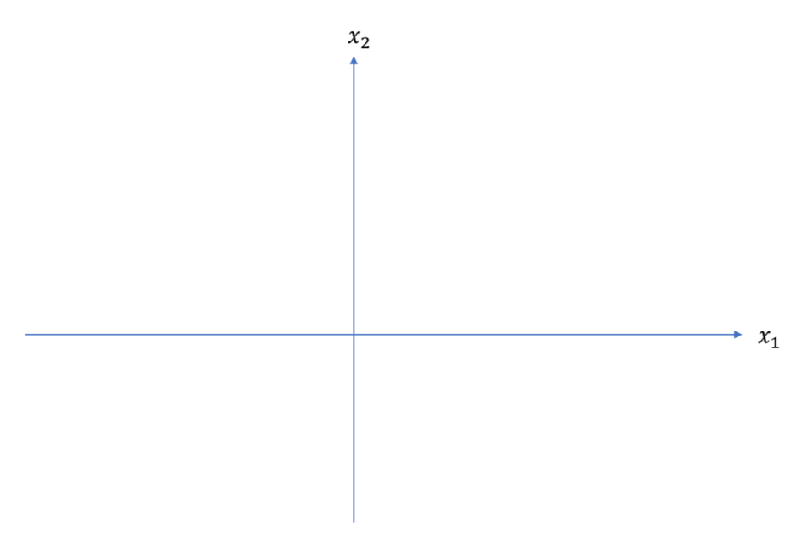
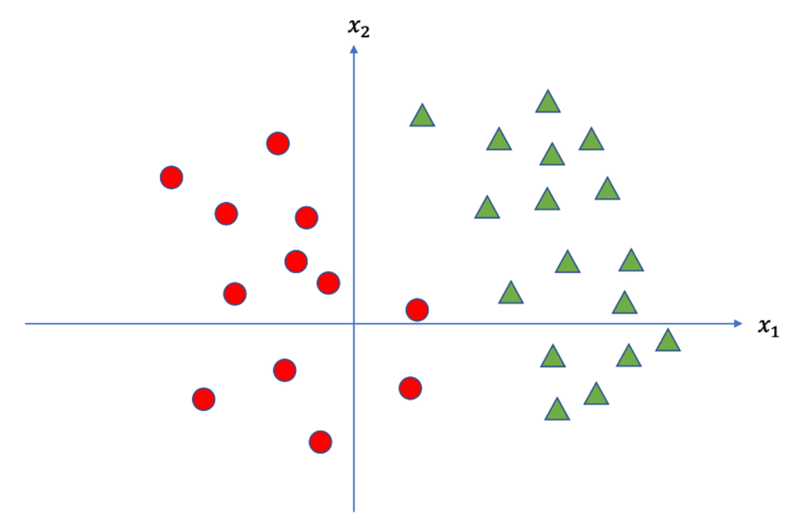
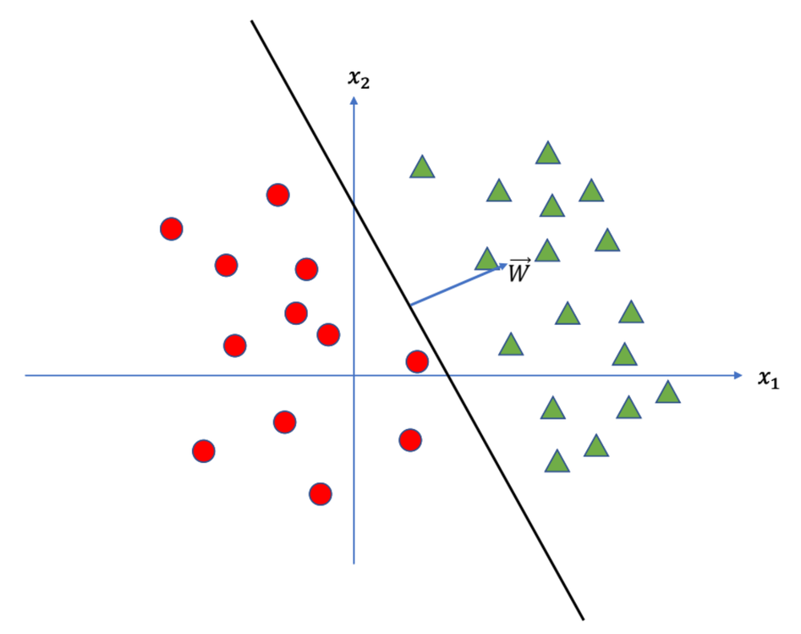
logistic-regression.pdf
terça-feira, 11 de julho de 2023 15:29

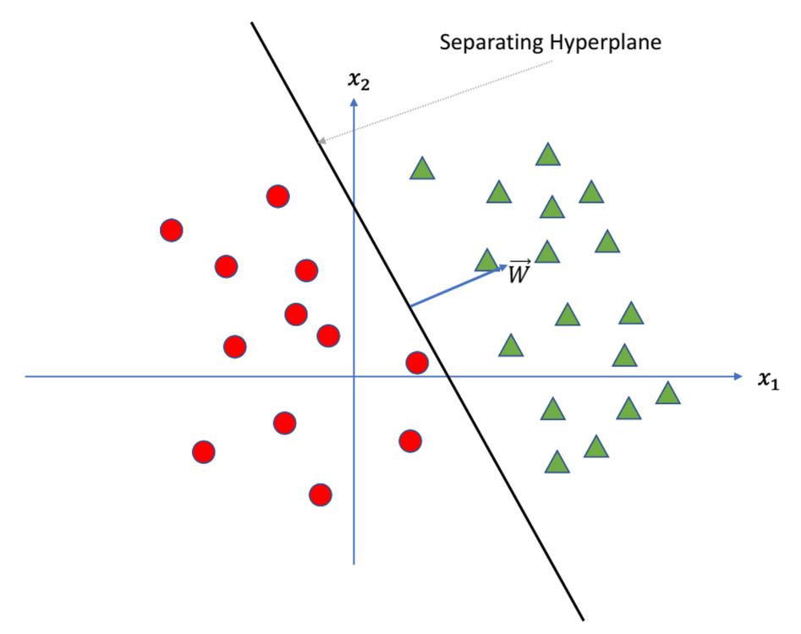
Logistic Regression

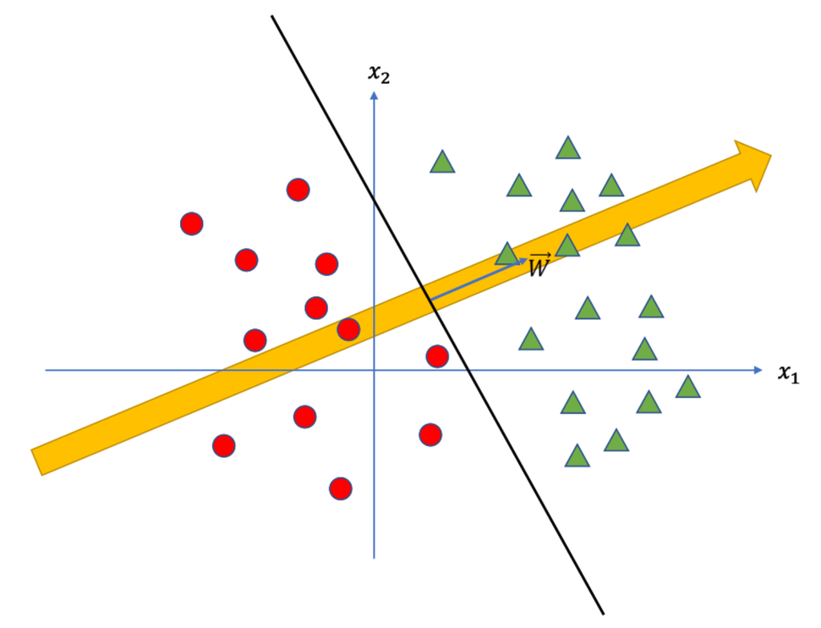
Motivation

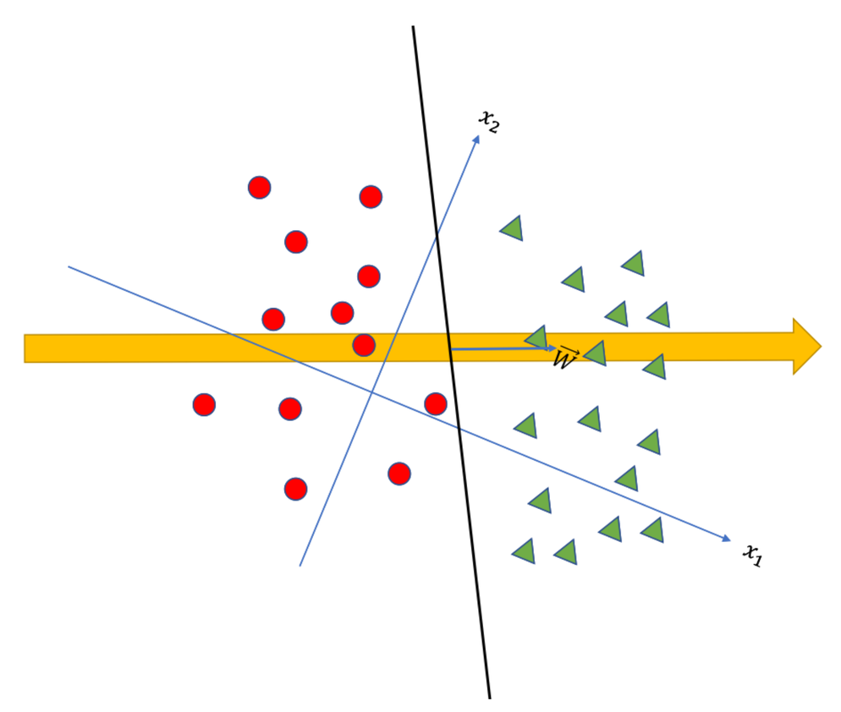












* Logistic Regression

A mesma ideia do perceptron de a partir de algebra, consome e ele você separa em grupos.

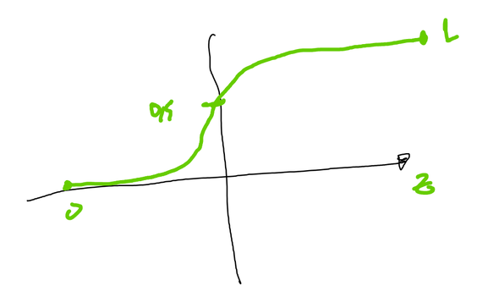
A base é: $Z = w^T \cdot X + b$ (pesos transpostos; input; bias (offset))

→ Porque w^T?
↳ Para que o resultado sejam valores p/ cada parametro

→ A partir disso usamos uma função de ativação. No caso da Logistic Regression é a função sigmoid

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

Ela vai gerar um valor entre 0 e 1, ao contrário da função sign que gerava apenas 1 e -1



Basicamente teremos que:

$$\hat{Y} = \text{Sigmoid}(w^T \cdot x + b)$$

(output predito)

→ os inputs não irão mudar, logo precisamos alterar os valores dos pesos e do bias para atingir os valores esperados

→ Para recalcularmos os valores de "w" e "b", temos que ter uma função de perda que indique quão distante do correto estamos

$$\mathcal{L}(a, y) = \mathcal{L}(\hat{Y}, Y)$$

(output predito — label correta)

No caso iremos usar o log loss, que é dada por:

$$\mathcal{L}(\hat{Y}, Y) = -(Y \log \hat{Y} + (1-Y)) \log(1-\hat{Y})$$

Ela é uma melhor opção para isso porque funciona melhor no gradiente descendente, por ser convexa.

→ A função de erro calcula para um único input, precisamos calcular a função de custo para todas as entradas

Custo dos parametros — Cost Function ⇒

$$J(w,b) = \frac{1}{m}\sum_{i=1}^{m} \mathcal{L}(\hat{Y}^{(i)}, Y^{(i)})$$

média de todas as loss function onde m = quantidade de inputs de X

Ao juntarmos as duas funções, temos que:

cost function + loss function →

$$J(w,b) = -\frac{1}{m}\sum_{i=1}^{n} \left[Y^{(i)} \log \hat{Y}^{(i)} + (1-Y^{(i)}) \log(1-\hat{Y}^{(i)}) \right]$$

A partir do custo, iremos buscar "w" e "b" que minimizem essa cost function e iremos fazer isso pelo gradiente descendente

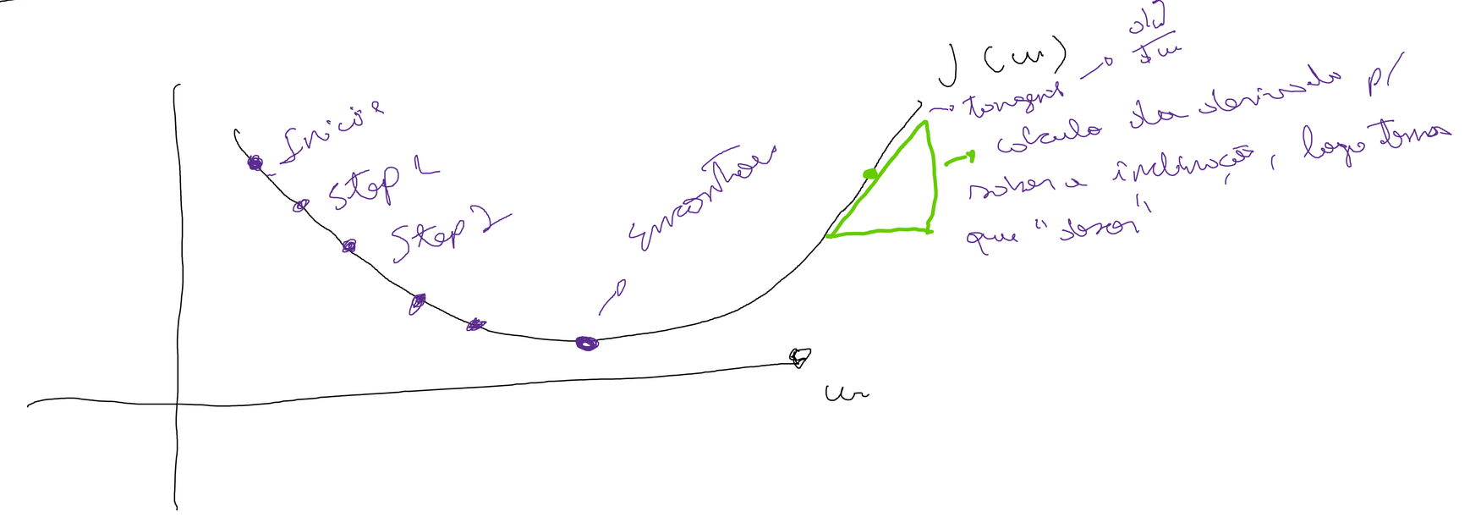
→ Como que você usa o gradiente descendente?

você inicializa os parametros ("w" e "b") com um valor aleatório calcula o gradiente e vai realizando os passos até chegar em mínimos locais satisfatórios

(a função é convexa do tipo ∪ logo, menos mínimos locais p/ se compreender)

↓

Para logistic regression o mais comum é inicializar com zero, mas pode usar inicialização aleatória



→ Step Function = Função que vai "andando" até o global optimal
(taxa de aprendizado)

* BlueBrown

- Backpropagation
  Algoritmo p/ computar o gradiente, já que precisamos dele para minimizar a cost function

Forward é o normal, gera o resultado
Back você vê se esse resultado tá distante por meio da loss function e calcula o gradiente descendente de cada camada

Pelo visto cada camada é o mesmo esquema de $Z = w^T x + b$

(→ usa a regra de cadeia das derivadas p/ checar o quão de peso influente uma variação em A vai ser no valor de B, onde A → ... → B)

→ Você repete o ... (e o backwards até ter um valor que tenha minimizado uma função

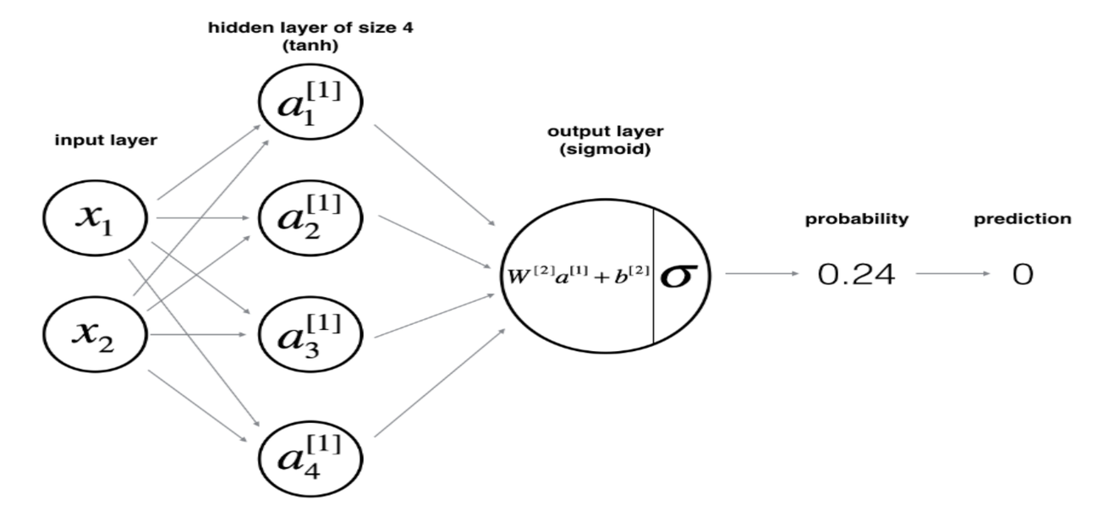
Faz o passo a passo comum. A diferença é que cada nova layer pode ter calculos diferentes mas vai usar a entrada da anterior e ser usado no input da proxima camada

* Passo a Passo

(Classificando não-linearidade de Planar classificatios)

O modelo contém uma hidden layer que vai usar a função tanh como função de ativação. O output usará sigmoid

Img



- 1 - Inicializa os parametros.
  ↳ nós temos duas variável X_1 e X_2, logo iremos precisar de w_1, w_2 e b como parametros
  ↳ w_1 tem que ter shape (size-hidden, size-input)

- 2 - Faz o forward propagation
  ↳ Para obter os primeiros resultados e saber o quão errado estão os pesos aleatórios
  ↳ Computamos Z e depois A de cada camada

  Nesse caso temos camada 1 = hidden, onde a função de ativação é (tanh), logo:

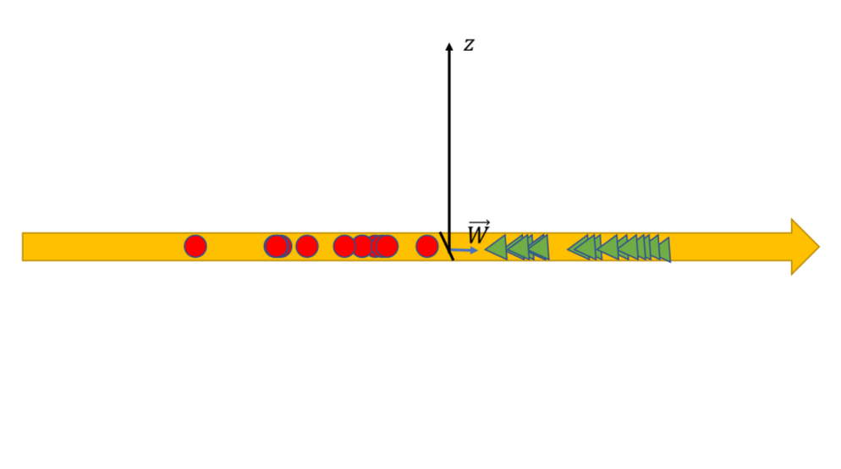
  $Z^{[1]} = W_1 \cdot X + b_1$
  $A^{[1]} = \tanh(Z^{[1]})$
  } → sempre referente a camada 1

  A camada 2 = output layer tem como entrada o resultado da 1 e usa o sigmoid como função de ativação

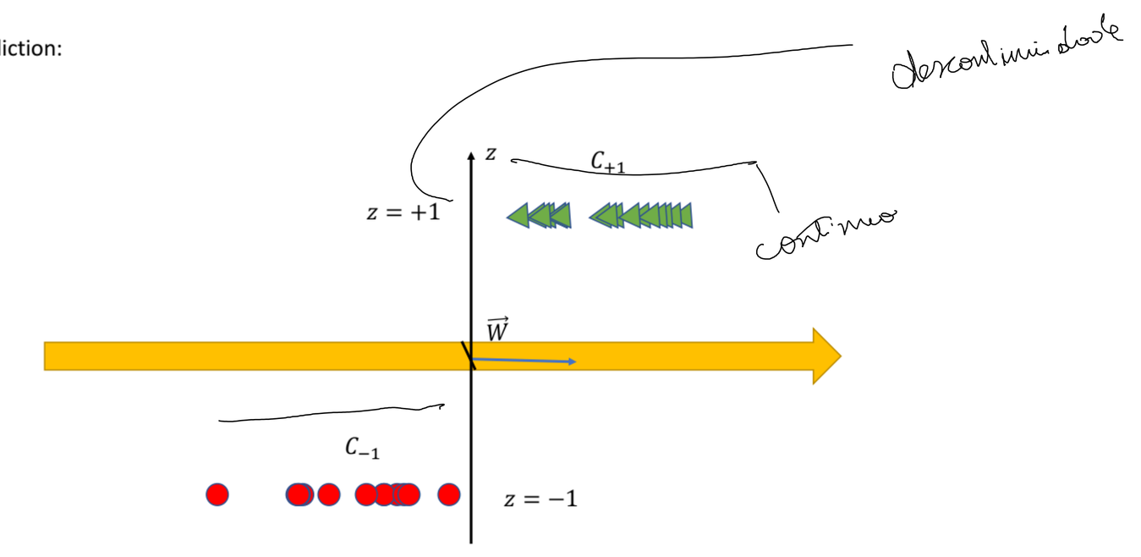
  $Z^{[2]} = W_2 \cdot A_1 + b_2$
  $A^{[2]} = \text{sigmoid}(Z^{[2]})$

  ↳ Guardamos esses valores, eles serão usados na backpropagation
  ↳ $A^{[2]}$ é o $\hat{Y}$, ou seja, a classificação feita pelo algoritmo

- 3 → Computar a Cost Function
  ↳ Temos o Y e os $\hat{Y}$ preditos, logo iremos calcular o custo p/ minimizarmos na prox etapa do gradiente
  ↳ Calculamos pela cross entropy loss
  (→ lembrando que $A^{[2]}$ é $\hat{Y}$, resultado)



Perceptron's prediction:

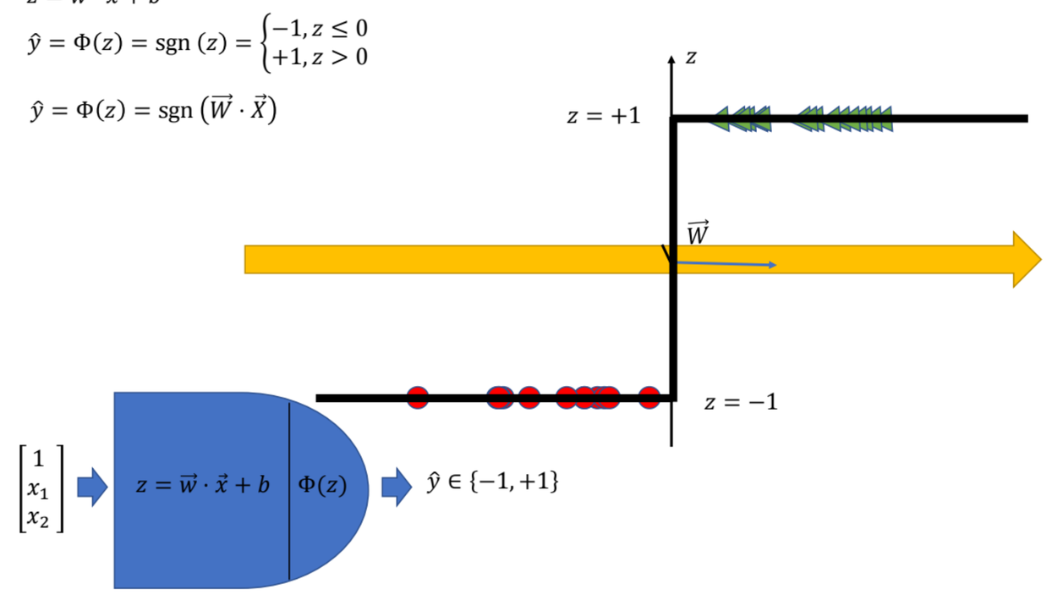


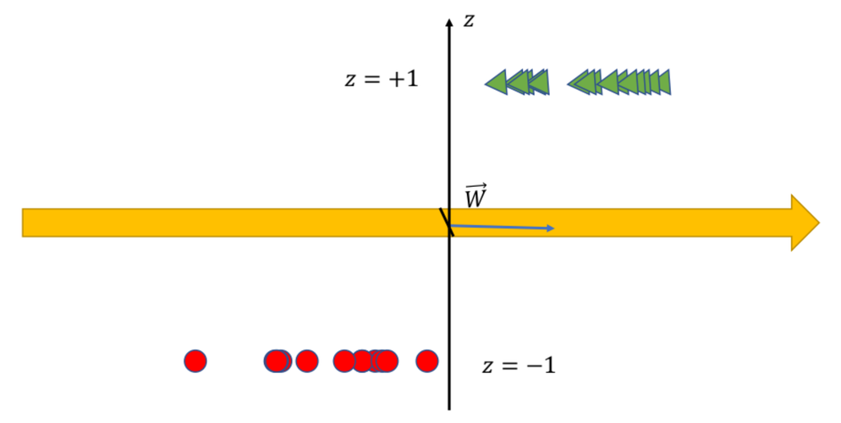
Perceptron's prediction:

$z = \vec{w} \cdot \vec{x} + b$

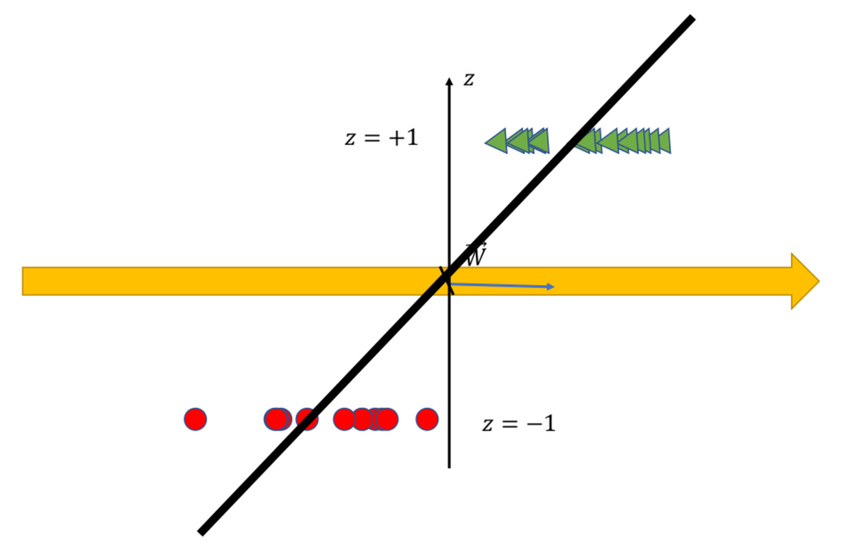
$\hat{y} = \Phi(z) = \mathrm{sgn}(z) = \begin{cases} -1, z \le 0 \\ +1, z > 0 \end{cases}$

$\hat{y} = \Phi(z) = \mathrm{sgn}(\vec{w} \cdot \vec{x})$

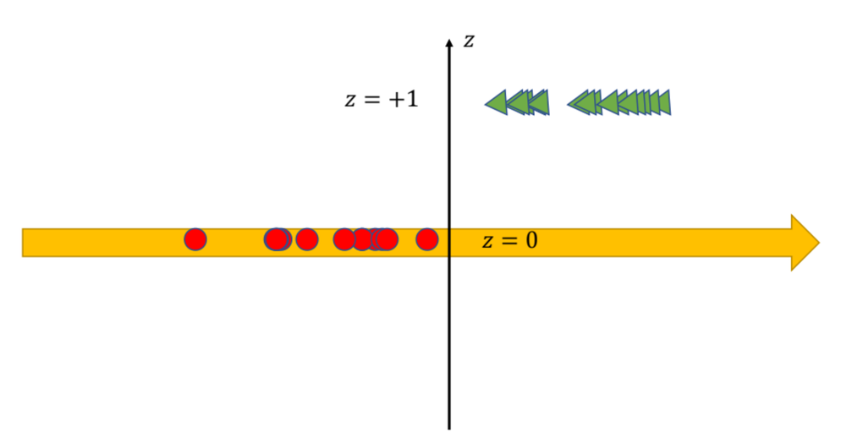




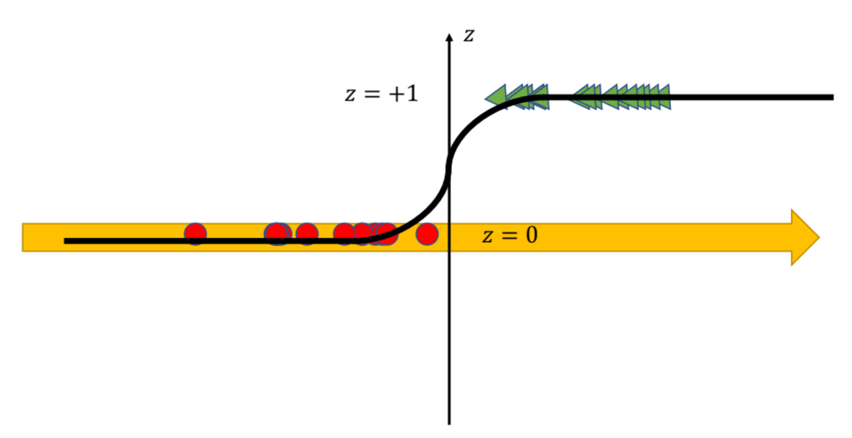
Linear regression?



Probabilistic view



Logistic Regression!



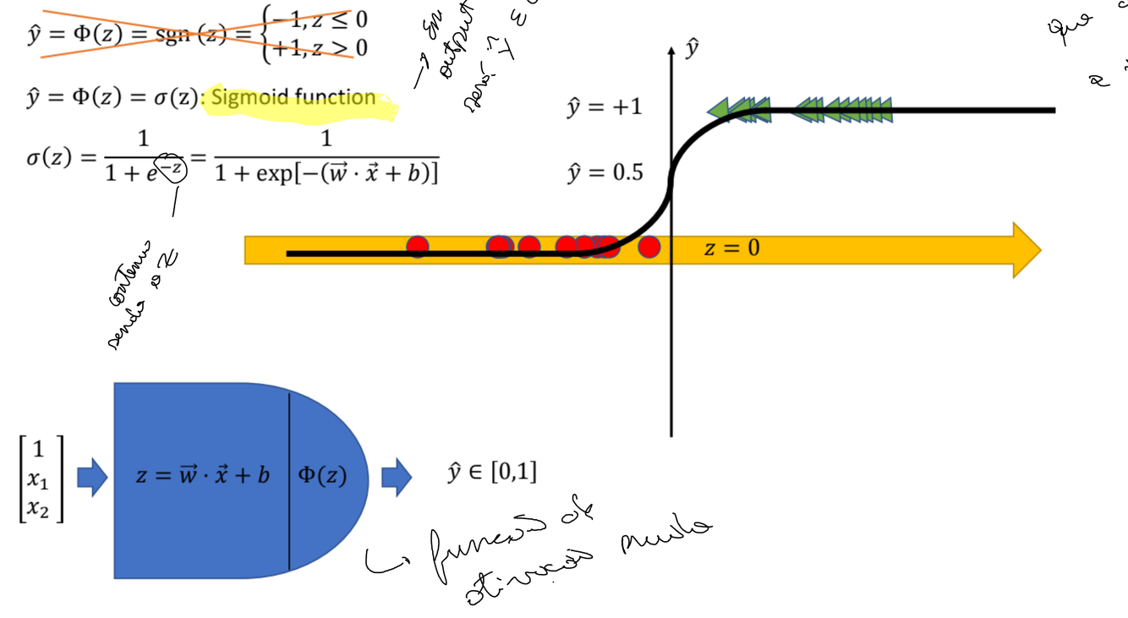
Logistic Regression!

$z = \vec{w} \cdot \vec{x} + b$

~~$\hat{y} = \Phi(z) = \mathrm{sgn}(z) = \begin{cases} -1, z \le 0 \\ +1, z > 0 \end{cases}$~~

$\hat{y} = \Phi(z) = \sigma(z)$: Sigmoid function

$\sigma(z) = \frac{1}{1+e^{-z}} = \frac{1}{1+\exp[-(\vec{w} \cdot \vec{x} + b)]}$



→ em vez de sign
o output ser
$\hat{y} \in [0,1]$

contínuo sendo e^{z}

↳ funções de ativação muito

continua tendo que achar $z = \vec{w} \cdot \vec{x} + b$
a diferença é na função de sign
e entra o sigmoid

Logistic Regression

Vai usar a função sigmoid em vez de sign
mas continua tendo que calcular o dot product + offset

Sigmoid tem como equação

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

↳ euler

$$w := w - \alpha \left(\frac{dJ(w,b)}{dw} \right)$$

→ taxa

"dw"

nome da variável → convenção

$$b := b - \alpha \left(\frac{dJ(w,b)}{db} \right)$$

"db"

esse dw é baseado no slide loss, pq derivadas indica a inclinação

A taxa de aprendizado vai gerir o tamanho do "passo"

Notação de derivada parcial

$$\frac{\partial J(w,b)}{\partial w}$$

→ Vídeo do Andrew Ng

Recap

$z = w^T x + b$

$\hat{y} = a = \sigma(z)$

Computational Graph

x_1, w_1, x_2, w_2, b → $z = w_1 x_1 + w_2 x_2 + b$ → $\hat{y} = \mathrm{sig}(z)$ → $\mathcal{L}(a, y)$

Precisamos modificar os parâmetros (w_1, w_2 e b) para minimizar o loss

→ Backprop

Queremos derivar voltando do loss para definir e variar os parâmetros

Começamos com $\frac{d\mathcal{L}(a,y)}{da}$ → "da" $= \frac{-y}{a} + \frac{1-y}{1-a}$

"voltando", teremos que $dz = \frac{d\mathcal{L}(a,y)}{dz} = a - y$

Depois compute "w" e "b" → $d_{w_1} = x_1 \cdot d_z$

$d_{w_2} = x_2 \cdot d_z$

$d_b = d_z$

→ O ponto de derivar é: Se pegamos um valor v e mudar, o que irá ocorrer com os subsequentes?

$\frac{dJ}{dv}$ → Se mudamos "v", o que acontece com "J"?

$v = a + u$ → $J = 3v$

→ Back propagation entra aí, porque você vai voltando no grafo para avaliar mudando o output

$v = a + u$ → $J = 3v$

Se eu mudar a, vai mudar v, que vai mudar J

a → v → J → Chain Rule das derivadas

→ O foco é achar a derivada do final output

→ O que é J? Cost function

$J(w,b) = \frac{1}{m} \sum_{i}^{} \mathcal{L}(a^{(i)}, y)$ | $a^{(i)} = \hat{y}^{(i)} = \mathrm{sig}(z) = \mathrm{sig}(w^T x^{(i)} + b)$

$J = \frac{1}{m} \sum_{i} y^{(i)} \log(A^{\ldots}) + (\ldots) \log(\ldots)$

• 4 - Back Propagation

↳ Você vai calculando as derivadas parciais, de modo que para cada parâmetro e 7 você calcula

↳ calcula dw_1, dw_2, db_2, db_1

Summary of gradient descent

$dz^{[2]} = a^{[2]} - y$	$dZ^{[2]} = A^{[2]} - Y$
$dW^{[2]} = dz^{[2]} a^{[1]^T}$	$dW^{[2]} = \frac{1}{m} dZ^{[2]} A^{[1]^T}$
$db^{[2]} = dz^{[2]}$	$db^{[2]} = \frac{1}{m} np.sum(dZ^{[2]}, axis = 1, keepdims = True)$
$dz^{[1]} = W^{[2]T} dz^{[2]} * g^{[1]\prime}(z^{[1]})$	$dZ^{[1]} = W^{[2]T} dZ^{[2]} * g^{[1]\prime}(Z^{[1]})$
$dW^{[1]} = dz^{[1]} x^T$	$dW^{[1]} = \frac{1}{m} dZ^{[1]} X^T$
$db^{[1]} = dz^{[1]}$	$db^{[1]} = \frac{1}{m} np.sum(dZ^{[1]}, axis = 1, keepdims = True)$

• 5 → Implementar o update rule

↳ Atualizamos os valores de "W" e "b", de modo a usar o gradiente

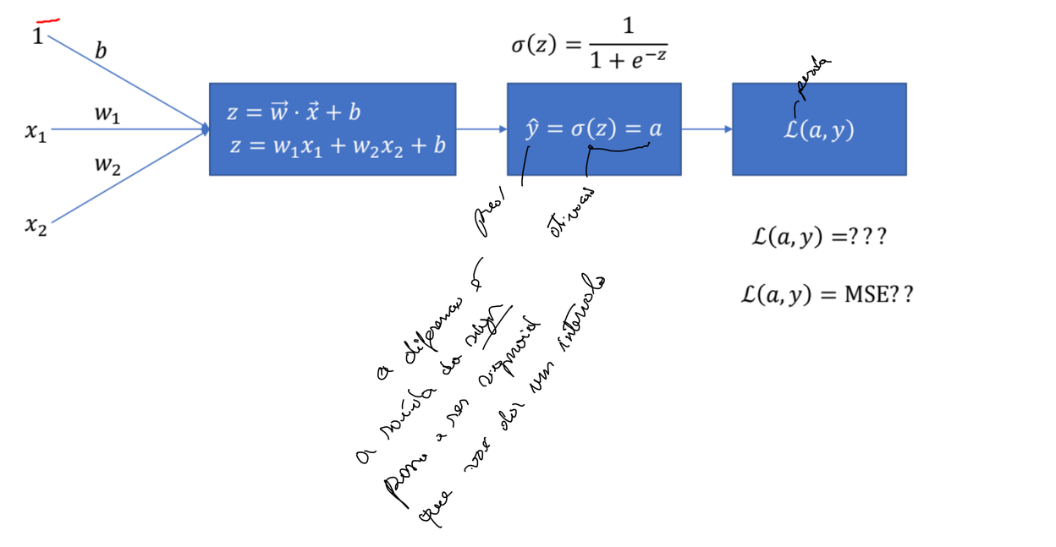
↳ A regra base é: $\theta = \theta - \alpha \frac{dJ}{d\theta}$ sendo θ = parâmetro, α = learning rate

↳ Ex: w_1 → $w_1 = w_1 - \alpha \cdot dw_1$

↳ vem do passo anterior

• 6 - Predições.

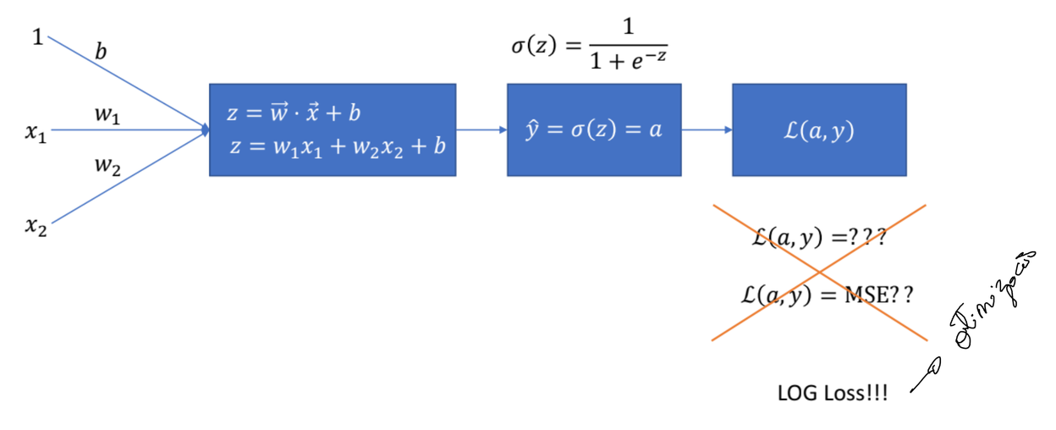
Training



→ outra diferença é que temos a função de perda/erro que vai recalcular, tipo aquela do W_i^N, no que iremos usar outra, não mais o MSE → median square error
iremos usar agora o log loss para realizar a otimização dos pesos

$\mathcal{L}(a, y)$ → esperado

$\hat{Y}$, resultado do sigmoid



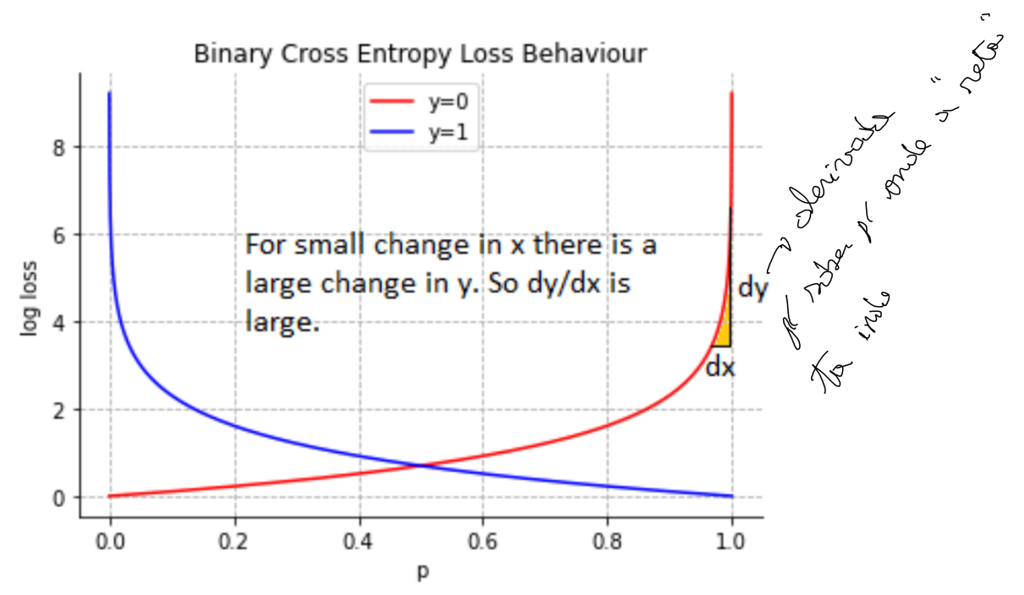
→ Formula de Log Loss

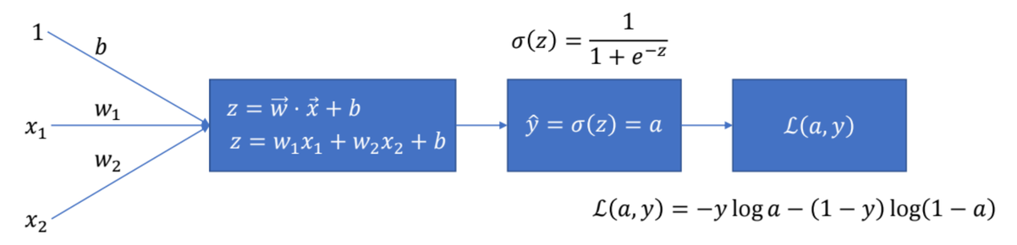
$\mathcal{L}(a,y) = -y \log a - (1-y) \log(1-a)$

esperado — $\hat{y}$ (predito)

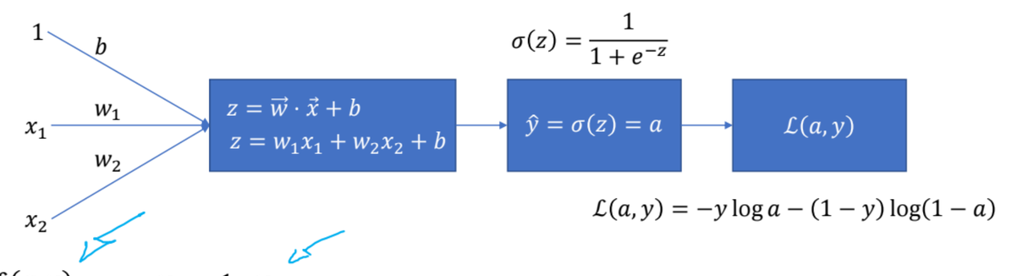
Log loss

$\mathcal{L}(a, y) = -y \log a - (1-y) \log(1-a)$





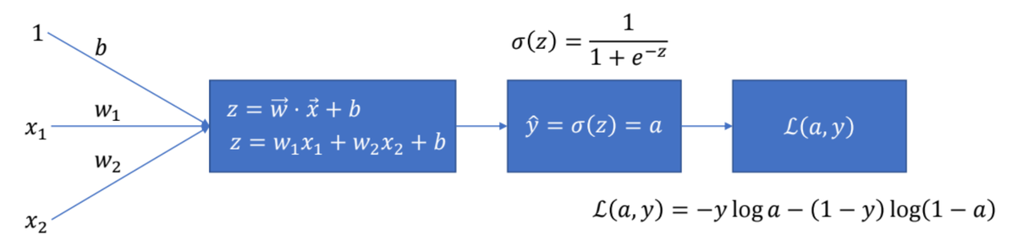
$\mathcal{L}(a, y) = -y \log a - (1-y) \log(1-a)$



$\mathcal{L}(a, y) = -y \log a - (1-y) \log(1-a)$

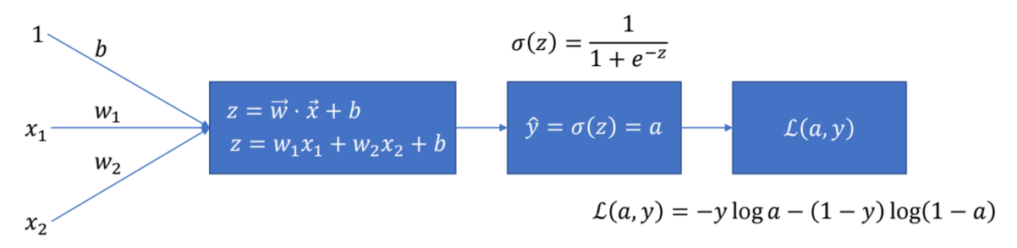
- $\frac{\partial \mathcal{L}(a,y)}{\partial a} = -\frac{y}{a} + \frac{1-y}{1-a}$

↳ Derivada parcial



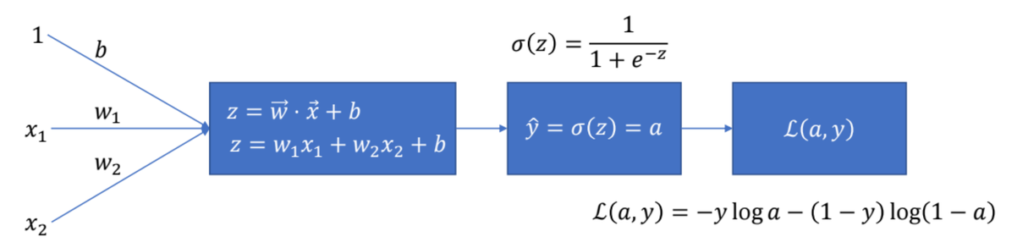
$\mathcal{L}(a, y) = -y \log a - (1-y) \log(1-a)$

- $\frac{\partial \mathcal{L}(a,y)}{\partial a} = -\frac{y}{a} + \frac{1-y}{1-a}$
- $\frac{\partial a(z)}{\partial z} = \frac{\partial}{\partial z}\left((1+e^{-z})^{-1}\right) = a[1-a]$



$\mathcal{L}(a, y) = -y \log a - (1-y) \log(1-a)$

- $\frac{\partial \mathcal{L}(a,y)}{\partial a} = -\frac{y}{a} + \frac{1-y}{1-a}$
- $\frac{\partial a(z)}{\partial z} = \frac{\partial}{\partial z}\left((1+e^{-z})^{-1}\right) = a[1-a]$
- $\frac{\partial \mathcal{L}(a,y)}{\partial z} = \frac{\partial \mathcal{L}(a,y)}{\partial a} \frac{\partial a(z)}{\partial z} = a - y$

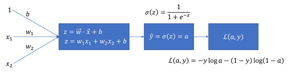


$\mathcal{L}(a, y) = -y \log a - (1-y) \log(1-a)$

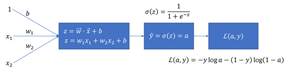
- $\frac{\partial \mathcal{L}(a,y)}{\partial a} = -\frac{y}{a} + \frac{1-y}{1-a}$
- $\frac{\partial a(z)}{\partial z} = \frac{\partial}{\partial z}\left((1+e^{-z})^{-1}\right) = a[1-a]$

$\frac{\partial z}{\partial z}$ $\frac{\partial z}{\partial z}$

- $\frac{\partial \mathcal{L}(a,y)}{\partial z} = \frac{\partial \mathcal{L}(a,y)}{\partial a}\frac{\partial a(z)}{\partial z} = a - y$
- $\frac{\partial \mathcal{L}}{\partial w_1} = \frac{\partial \mathcal{L}}{\partial z}\frac{\partial z}{\partial w_1} = (a-y)x_1$

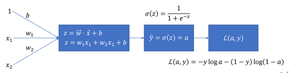


- $\frac{\partial \mathcal{L}(a,y)}{\partial a} = -\frac{y}{a} + \frac{1-y}{1-a}$
- $\frac{\partial a(z)}{\partial z} = \frac{\partial}{\partial z}\left((1+e^{-z})^{-1}\right) = a[1-a]$
- $\frac{\partial \mathcal{L}(a,y)}{\partial z} = \frac{\partial \mathcal{L}(a,y)}{\partial a}\frac{\partial a(z)}{\partial z} = a - y$
- $\frac{\partial \mathcal{L}}{\partial w_1} = \frac{\partial \mathcal{L}}{\partial z}\frac{\partial z}{\partial w_1} = (a-y)x_1$
- $\frac{\partial \mathcal{L}}{\partial w_2} = \frac{\partial \mathcal{L}}{\partial z}\frac{\partial z}{\partial w_2} = (a-y)x_2$



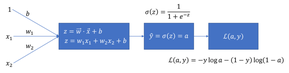
- $\frac{\partial \mathcal{L}(a,y)}{\partial a} = -\frac{y}{a} + \frac{1-y}{1-a}$
- $\frac{\partial a(z)}{\partial z} = \frac{\partial}{\partial z}\left((1+e^{-z})^{-1}\right) = a[1-a]$
- $\frac{\partial \mathcal{L}(a,y)}{\partial z} = \frac{\partial \mathcal{L}(a,y)}{\partial a}\frac{\partial a(z)}{\partial z} = a - y$
- $\boldsymbol{\frac{\partial \mathcal{L}}{\partial w_1} = \frac{\partial \mathcal{L}}{\partial z}\frac{\partial z}{\partial w_1} = (a-y)x_1}$
- $\boldsymbol{\frac{\partial \mathcal{L}}{\partial w_2} = \frac{\partial \mathcal{L}}{\partial z}\frac{\partial z}{\partial w_2} = (a-y)x_2}$
- $\boldsymbol{\frac{\partial \mathcal{L}}{\partial b} = \frac{\partial \mathcal{L}}{\partial z}\frac{\partial z}{\partial b} = (a-y)}$

Backpropagation



- $\frac{\partial \mathcal{L}(a,y)}{\partial a} = -\frac{y}{a} + \frac{1-y}{1-a}$
- $\frac{\partial a(z)}{\partial z} = \frac{\partial}{\partial z}\left((1+e^{-z})^{-1}\right) = a[1-a]$
- $\frac{\partial \mathcal{L}(a,y)}{\partial z} = \frac{\partial \mathcal{L}(a,y)}{\partial a}\frac{\partial a(z)}{\partial z} = a - y$
- $\boldsymbol{\frac{\partial \mathcal{L}}{\partial w_1} = \frac{\partial \mathcal{L}}{\partial z}\frac{\partial z}{\partial w_1} = (a-y)x_1}$
- $\boldsymbol{\frac{\partial \mathcal{L}}{\partial w_2} = \frac{\partial \mathcal{L}}{\partial z}\frac{\partial z}{\partial w_2} = (a-y)x_2}$
- $\boldsymbol{\frac{\partial \mathcal{L}}{\partial b} = \frac{\partial \mathcal{L}}{\partial z}\frac{\partial z}{\partial b} = (a-y)}$

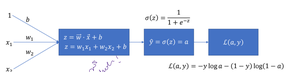
- $w_{1,i} \leftarrow w_{1,i-1} - \eta\,\vec{\nabla}_{\vec{w}}\mathcal{L}(\boldsymbol{a},\boldsymbol{y})$
- $w_{2,i} \leftarrow w_{2,i-1} - \eta\vec{\nabla}_{\vec{w}}\mathcal{L}(\boldsymbol{a},\boldsymbol{y})$
- $b_i \leftarrow b_{i-1} - \eta\vec{\nabla}_{\vec{w}}\mathcal{L}(\boldsymbol{a},\boldsymbol{y})$



- $\frac{\partial \mathcal{L}(a,y)}{\partial a} = -\frac{y}{a} + \frac{1-y}{1-a}$
- $\frac{\partial a(z)}{\partial z} = \frac{\partial}{\partial z}\left((1+e^{-z})^{-1}\right) = a[1-a]$
- $\frac{\partial \mathcal{L}(a,y)}{\partial z} = \frac{\partial \mathcal{L}(a,y)}{\partial a}\frac{\partial a(z)}{\partial z} = a - y$
- $\boldsymbol{\frac{\partial \mathcal{L}}{\partial w_1} = \frac{\partial \mathcal{L}}{\partial z}\frac{\partial z}{\partial w_1} = (a-y)x_1}$
- $\boldsymbol{\frac{\partial \mathcal{L}}{\partial w_2} = \frac{\partial \mathcal{L}}{\partial z}\frac{\partial z}{\partial w_2} = (a-y)x_2}$
- $\boldsymbol{\frac{\partial \mathcal{L}}{\partial b} = \frac{\partial \mathcal{L}}{\partial z}\frac{\partial z}{\partial b} = (a-y)}$

- $w_{1,i} \leftarrow w_{1,i-1} - \eta(\boldsymbol{a-y})\boldsymbol{x_1}$
- $w_{2,i} \leftarrow w_{2,i-1} - \eta(\boldsymbol{a-y})\boldsymbol{x_2}$
- $b_i \leftarrow b_{i-1} - \eta(\boldsymbol{a-y})$

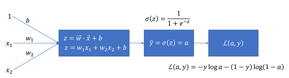
This is the loss related to 1 sample!



- Cost of *m* samples:

$$P(\cdot) = \prod_{i=1}^{m} P\left(y^{(i)}|\vec{x}^{(i)}\right) \quad \text{(Maximum Likelihood Estimation)}$$

- The MLE estimates the ser of parameters $\vec{\theta} = (\vec{w}, b)$ so that they are as close as possible to the real set of parameters $\vec{\theta}$ that generated such data samples by minimizing the loss $\mathcal{L}(y, \hat{y})$.



- Cost of *m* samples:

$$\log P(\cdot) = \log \prod_{i=1}^{m} P\left(y^{(i)}|\vec{x}^{(i)}\right)$$

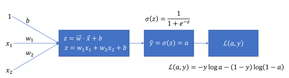
$$\log P(\cdot) = \sum_{i=1}^{m} \log P\left(y^{(i)}|\vec{x}^{(i)}\right)$$

$$\log P(\cdot) = -\sum_{i=1}^{m} \mathcal{L}\left(y^{(i)}|\vec{x}^{(i)}\right)$$

- Cost:

$$J(\vec{w}, b) = \frac{1}{m}\sum_{i=1}^{m} \mathcal{L}\left(y^{(i)}, \hat{y}^{(i)}\right)$$

Minimize the cost =
= Maximize the likelihood.



- Objective:
  - Find $\vec{w}, b$ that minimize

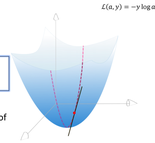
$$J(\vec{w}, b) = \frac{1}{m}\sum_{i=1}^{m} \mathcal{L}\left(y^{(i)}, \hat{y}^{(i)}\right)$$

$$\mathcal{L}(a, y) = -y \log a - (1 - y) \log(1 - a)$$

- Objective:
  - Find $\vec{w}, b$ that minimize

$$J(\vec{w}, b) = \sum \mathcal{L}(y, \hat{y})$$

1. Start at any point.
2. Walk in the direction of steepest descent, ie.,

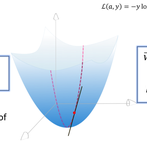


$$\mathcal{L}(a, y) = -y \log a - (1 - y) \log(1 - a)$$

- Objective:
  - Find $\vec{w}, b$ that minimize

$$J(\vec{w}, b) = \sum \mathcal{L}(y, \hat{y})$$

1. Start at any point.
2. Walk in the direction of steepest descent, ie.,



$$\vec{w} \leftarrow \vec{w} - \alpha \vec{\nabla}_w J(\vec{w}, b)$$

$$b \leftarrow b - \alpha \frac{d}{db} J(\vec{w}, b)$$

Notation

- Single Training Sample: $(\vec{x}, y)$, where:
  - $\vec{x} \in \Re^{n_x}$.
  - $y \in \{0,1\}$.
- m training samples: $\{(\vec{x}^{(1)}, y^{(1)}), (\vec{x}^{(2)}, y^{(2)}), \ldots, (\vec{x}^{(m)}, y^{(m)})\} = \{\vec{x}^{(i)}, y^{(i)}\}_{i=1}^{m}$, where:
  - m_{train} = Number of training samples.
  - m_{test} = Number of test samples.

- Single Training Sample: $(\vec{x}, y)$, where:
  - $\vec{x} \in \Re^{n_x}$.
  - $y \in \{0,1\}$.
- m training samples: $\{(\vec{x}^{(1)}, y^{(1)}), (\vec{x}^{(2)}, y^{(2)}), \ldots, (\vec{x}^{(m)}, y^{(m)})\} = \{\vec{x}^{(i)}, y^{(i)}\}_{i=1}^{m}$, where:
  - m_{train} = Number of training samples.
  - m_{test} = Number of test samples.

- $X = \begin{bmatrix} | & | & & | \\ \vec{x}^{(1)} & \vec{x}^{(2)} & \ldots & \vec{x}^{(m)} \\ | & | & & | \end{bmatrix}_{n_x}$ $\qquad Y = \begin{bmatrix} y^{(1)} & y^{(2)} & \ldots & y^{(m)} \end{bmatrix}$
- $X \in \Re^{(n_x \times m)}$, i.e., `X.shape=(n_x, m)`.
- $Y \in \Re^{(1 \times m)}$, i.e., `Y.shape = (1, m)`.

Gradient Descent on Logistic Regression

Forward Propagation for m examples

$$[w_1 \quad w_2 \quad \cdots \quad w_{n_x}] \begin{bmatrix} | & | & & | \\ \vec{x}^{(1)} & \vec{x}^{(2)} & \ldots & \vec{x}^{(m)} \\ | & | & & | \end{bmatrix} + [b \quad b \quad \cdots \quad b] =$$

Forward Propagation for m examples

Forward Propagation for m examples

$$[w_1 \quad w_2 \quad \cdots \quad w_{n_x}] \begin{bmatrix} | & | & & | \\ \vec{x}^{(1)} & \vec{x}^{(2)} & \dots & \vec{x}^{(m)} \\ | & | & & | \end{bmatrix} + [b \quad b \quad \cdots \quad b] =$$

$$= [z^{(1)} \quad z^{(2)} \quad \dots \quad z^{(m)}]$$

Forward Propagation for m examples

$$[w_1 \quad w_2 \quad \cdots \quad w_{n_x}]_{1\times n_x} \begin{bmatrix} | & | & & | \\ \vec{x}^{(1)} & \vec{x}^{(2)} & \dots & \vec{x}^{(m)} \\ | & | & & | \end{bmatrix}_{n_x \times m} + [b \quad b \quad \cdots \quad b]_{1\times m} =$$

$$= [z^{(1)} \quad z^{(2)} \quad \dots \quad z^{(m)}]$$

Forward Propagation for m examples

$$[w_1 \quad w_2 \quad \cdots \quad w_{n_x}]_{1\times n_x} \begin{bmatrix} | & | & & | \\ \vec{x}^{(1)} & \vec{x}^{(2)} & \dots & \vec{x}^{(m)} \\ | & | & & | \end{bmatrix}_{n_x \times m} + [b \quad b \quad \cdots \quad b]_{1\times m} =$$

$$= Z = [z^{(1)} \quad z^{(2)} \quad \dots \quad z^{(m)}]$$

Forward Propagation for m examples

$$[w_1 \quad w_2 \quad \cdots \quad w_{n_x}]_{1\times n_x} \begin{bmatrix} | & | & & | \\ \vec{x}^{(1)} & \vec{x}^{(2)} & \dots & \vec{x}^{(m)} \\ | & | & & | \end{bmatrix}_{n_x \times m} + [b \quad b \quad \cdots \quad b]_{1\times m} =$$

$$= Z = [z^{(1)} \quad z^{(2)} \quad \dots \quad z^{(m)}]$$

$$A\sigma(Z) = [a^{(1)} \quad a^{(2)} \quad \dots \quad a^{(m)}]$$

$\mathcal{L}(a, y)$

$$A = [y^{(1)} \quad y^{(2)} \quad \dots \quad y^{(m)}]$$

Forward Propagation for m examples

$$[w_1 \quad w_2 \quad \cdots \quad w_{n_x}]_{1\times n_x} \begin{bmatrix} | & | & & | \\ \vec{x}^{(1)} & \vec{x}^{(2)} & \dots & \vec{x}^{(m)} \\ | & | & & | \end{bmatrix}_{n_x \times m} + [b \quad b \quad \cdots \quad b]_{1\times m} =$$

$$= Z = [z^{(1)} \quad z^{(2)} \quad \dots \quad z^{(m)}]$$

$$A\sigma(Z) = [a^{(1)} \quad a^{(2)} \quad \dots \quad a^{(m)}]$$

$\mathcal{L}(a, y)$

$$A = [y^{(1)} \quad y^{(2)} \quad \dots \quad y^{(m)}]$$

$$\frac{\partial \mathcal{L}(a,y)}{\partial z} = \frac{\partial \mathcal{L}(a,y)}{\partial a} \frac{\partial a(z)}{\partial z} = a - y$$

$$dZ = A - Y = [a^{(1)} - y^{(1)} \quad a^{(2)} - y^{(2)} \quad \dots \quad a^{(m)} - y^{(m)}]$$

$$\mathcal{L}(a, y) = -y \log a - (1 - y) \log(1 - a)$$

Recall the gradient descent for 1 sample:

- $\frac{\partial \mathcal{L}}{\partial w_1} = \frac{\partial \mathcal{L}}{\partial z} \frac{\partial z}{\partial w_1} = (\boldsymbol{a - y})\boldsymbol{x_1}$
- $\frac{\partial \mathcal{L}}{\partial w_2} = \frac{\partial \mathcal{L}}{\partial z} \frac{\partial z}{\partial w_2} = (\boldsymbol{a - y})\boldsymbol{x_2}$
- $\frac{\partial \mathcal{L}}{\partial b} = \frac{\partial \mathcal{L}}{\partial z} \frac{\partial z}{\partial b} = (\boldsymbol{a - y})$

- $w_{1,i} \leftarrow w_{1,i-1} - \eta(\boldsymbol{a - y})\boldsymbol{x_1}$
- $w_{2,i} \leftarrow w_{2,i-1} - \eta(\boldsymbol{a - y})\boldsymbol{x_2}$
- $b_i \leftarrow b_{i-1} - \eta(\boldsymbol{a - y})$

This is the loss related to 1 sample!

$$\mathcal{L}(a, y) = -y \log a - (1 - y) \log(1 - a)$$

Gradient descent for m samples:

$$J(\vec{w}, b) = \frac{1}{m} \sum_{i=1}^{m} \mathcal{L}(y^{(i)}, \hat{y}^{(i)})$$

$$\frac{\partial}{\partial w_k} J(\vec{w}, b) = \frac{1}{m} \sum_{i=1}^{m} \mathcal{L}(y^{(i)}, \hat{y}^{(i)})$$

Update:

$$\left[w_k \leftarrow w_k - \eta \frac{\partial}{\partial w_k} J(\vec{w}, b) \right]_{k=1}^{n_x}$$

$$b \leftarrow b - \eta \frac{\partial}{\partial b} J(\vec{w}, b)$$

Thank you