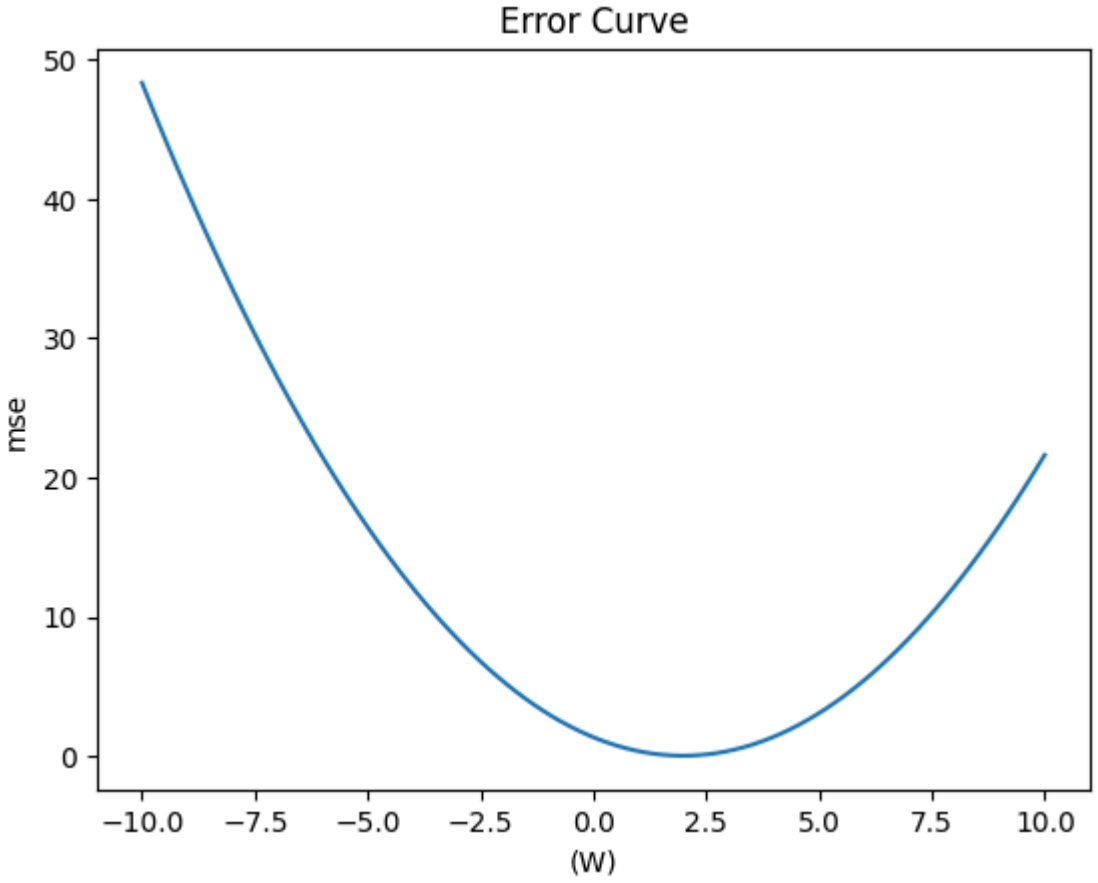
Simone Truglia:

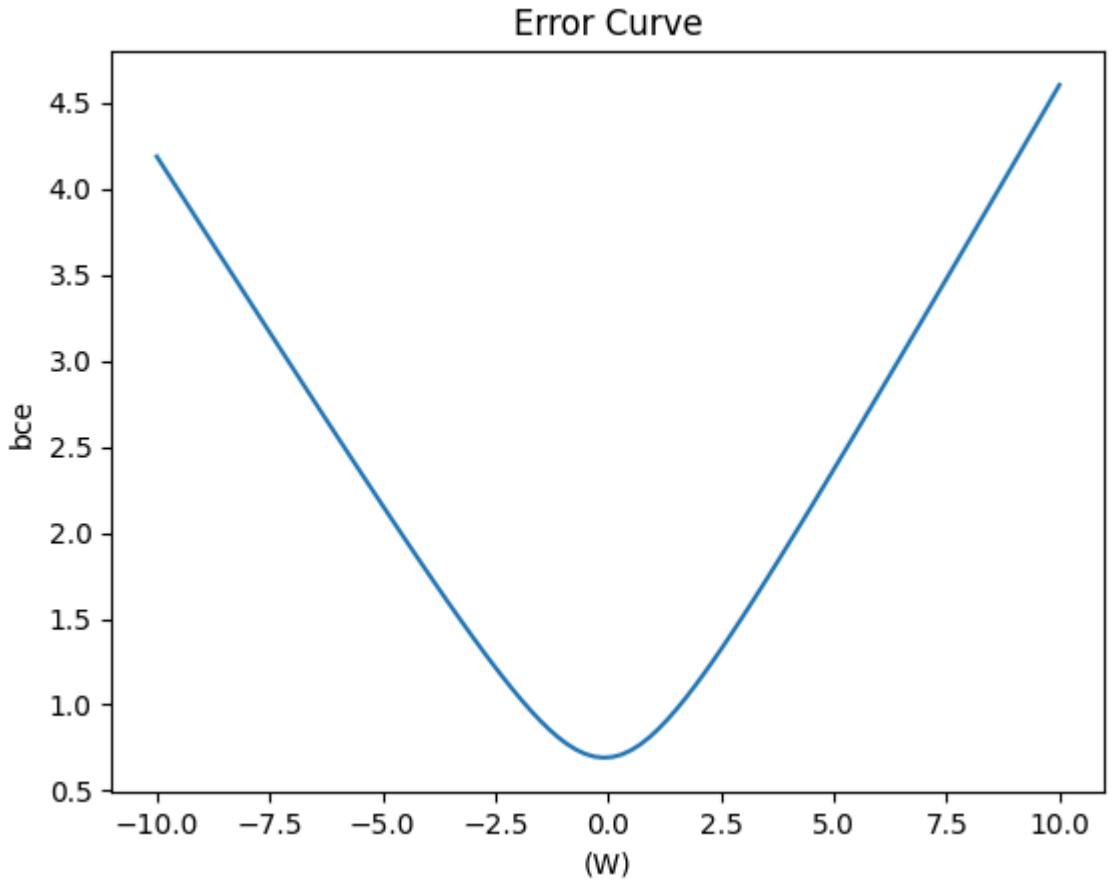
Enzo, sto cercando di capire come varia la funzione di costo, in relazione al problema specifico e alla funzione di attivazione utilizzata.

Ho fatto un notebook (https://colab.research.google.com/drive/1uuQ3Ws9tn3KBEri-QL98LBKda1ed9-PZ?usp=sharing) che sulla base di 3 paramentri, crea un dataset randomico allineato a quel tipo di problema (esempio una distribuizione per la regressione lineare e una serie di zeri e uni per la classificazione binaria) e mi plotta la funzione di costo scelta in relazione ai pesi, con la funzione di attivazione selezionata, in base al problema I paramentri possibili, al momento, sono: problem = 'linear' **OPPURE** 'binary_classification' loss = 'mse' **OPPURE** 'bce' activation = 'linear' **OPPURE** 'bce

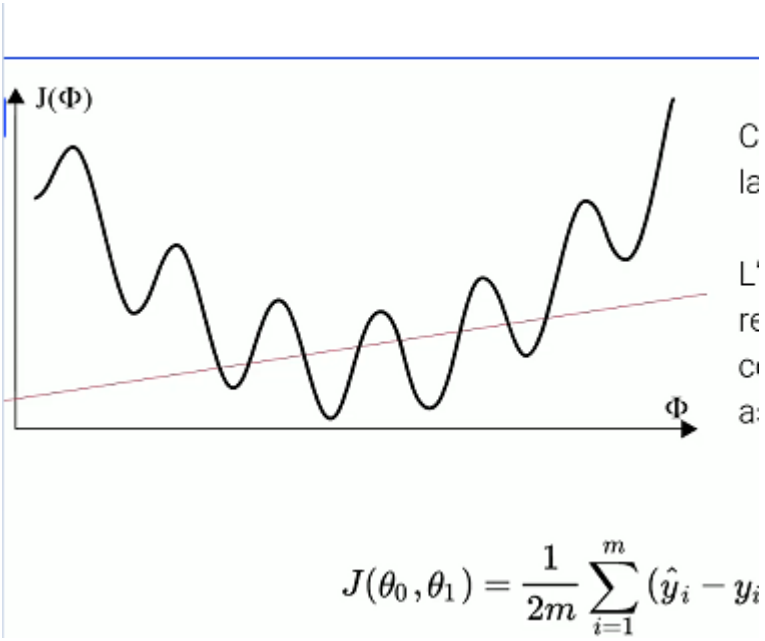
Ora, se provo a plottare un problema '**linear**' con loss '**mse**' e una attivazione '**linear**', ottengo quello che mi aspetto, una parabola convessa



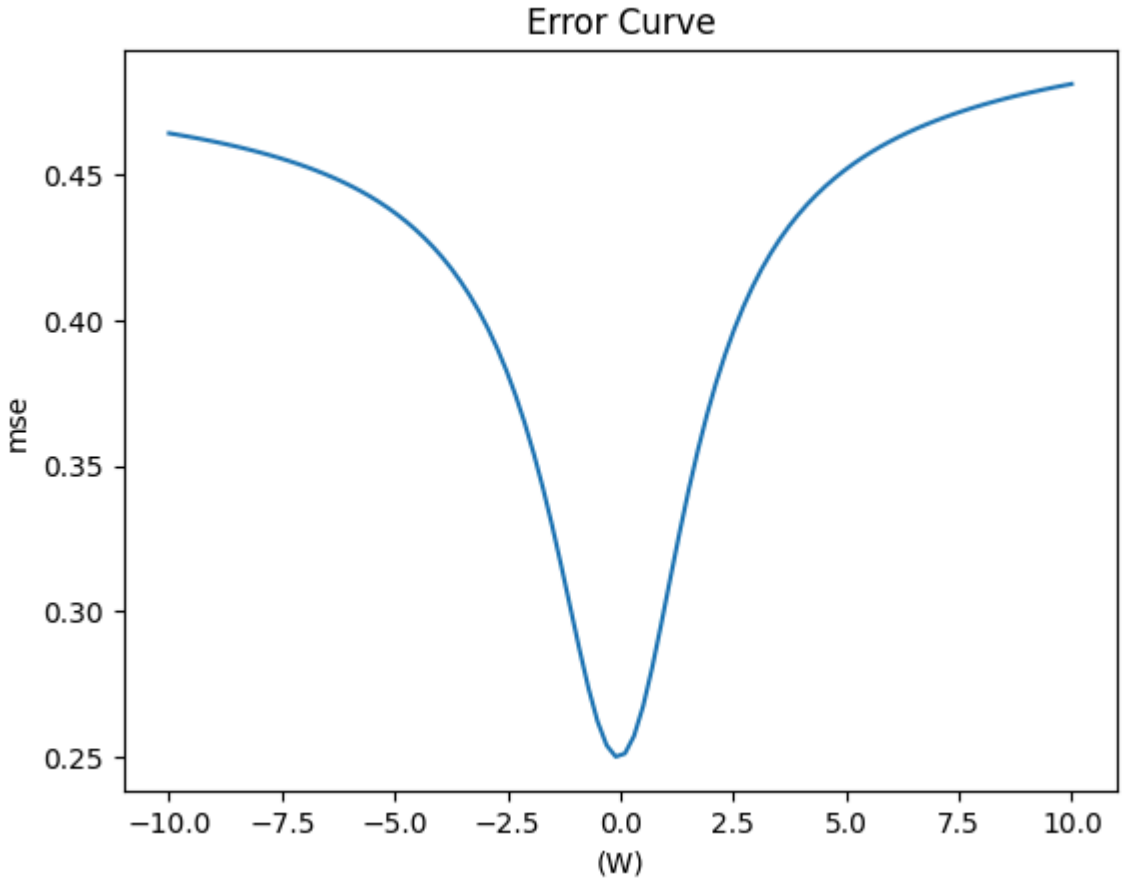
Se invece provo a plottare un problema '**binary_classification**' con loss '**bce**' e una attivazione '**sigmoid**', anche qui ottendo quello che mi aspetto, la funzione data dalla somma dei due logartimi



Allora ho provato a plottare un problema 'binary_classification' con loss 'mse' e una attivazione 'sigmoid', e mi aspettavo di vedere un grafico simile a questo che c'è nelle slides della lezione sulla prima lezione della regressione logistica:



e invece continuo a ottenere questo qui:



ho scritto tipo 10 codici diversi a mano, prendendo vie diverse, e ho ottenuto lo stesso grafico

(p.s. su tutti i grafici per semplicità **sto considerando un solo peso e un b = 0**)

Il ragionamento torna, perché abbiamo che sia limite che tende a +inf che il limite che tende a -inf della derivata sono uguali a zero, il che torna con i due asintoti orizzontali del grafico che ottengo io

Let's assume that we have a simple neural network with weights θ such as $z = \theta^T x$, and outputs $\hat{y} = \sigma(z)$ after a sigmoid activation.

The chain rule gives us the gradient of the loss L with respect to the weights θ:

$$\frac{\partial L}{\partial \theta} = \frac{\partial L}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial z} \frac{\partial z}{\partial \theta}$$

MSE loss is expressed as follows:

$$L(y, \hat{y}) = \frac{1}{2}(y - \hat{y})^2$$

Thus, the gradient with respect to θ is:

$$\begin{aligned} \frac{\partial L}{\partial \theta} &= -(y - \hat{y})\sigma(z)(1 - \sigma(z))x \\ &= -(y - \hat{y})\hat{y}(1 - \hat{y})x \end{aligned}$$

We can see that $\sigma(z)(1 - \sigma(z))$ makes the gradient vanish if $\sigma(z)$ is too close to 0 or 1. Thus, the neural net can't train properly.

(https://github.com/Jonas1312/mse-for-binary-classification)

Enzo:

Simone hai provato a scrivere il codice utilizzando i due parametri (w e b) previsti dalla regressione logistica?

Facendolo:

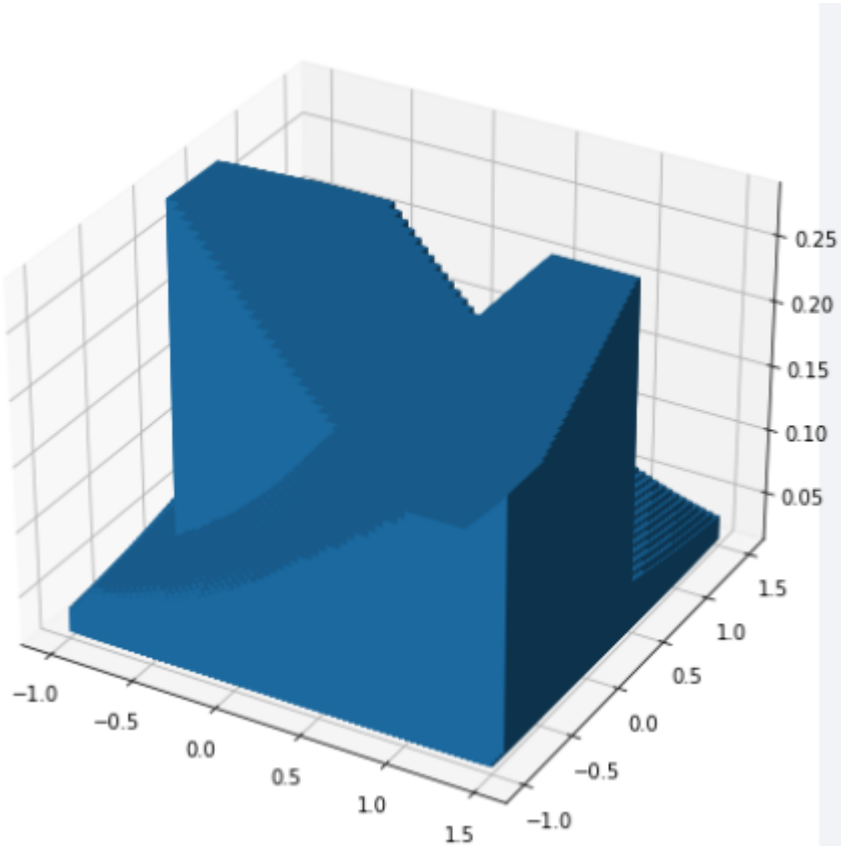
```
theta_0 = np.linspace(-1, 1, 50)
theta_1 = np.linspace(-1, 1, 50)
_theta0, _theta1 = np.meshgrid(theta_0, theta_1)
theta_0, theta_1 = _theta0.ravel(), _theta1.ravel()
errs = np.zeros_like(theta_0)
x = np.linspace(0, 1, len(theta_0))
y = np.zeros_like(theta_0)
y[int(len(y) / 2):] = 1

for i in range(len(theta_0)):
    for o, v in enumerate(x):
        y_pred = 1 / (1 + np.exp(-(theta_1[i] * v + theta_0[i])))
        y_true = y[i]
        errs[i] += ((y_true - y_pred) ** 2)
    errs[i] /= (2 * len(x))
```

```
fig = plt.figure(figsize=(8, 8))
ax1 = fig.add_subplot(111, projection='3d')
ax1.bar3d(theta_0, theta_1, np.min(errs), .5, .5, errs)
plt.show()
```

(codice schifoso, solo per provare velocemente)

otteniamo il grafico che ti metto di seguito, che ha 2 minimi, uno "a destra" e uno "a sinistra"

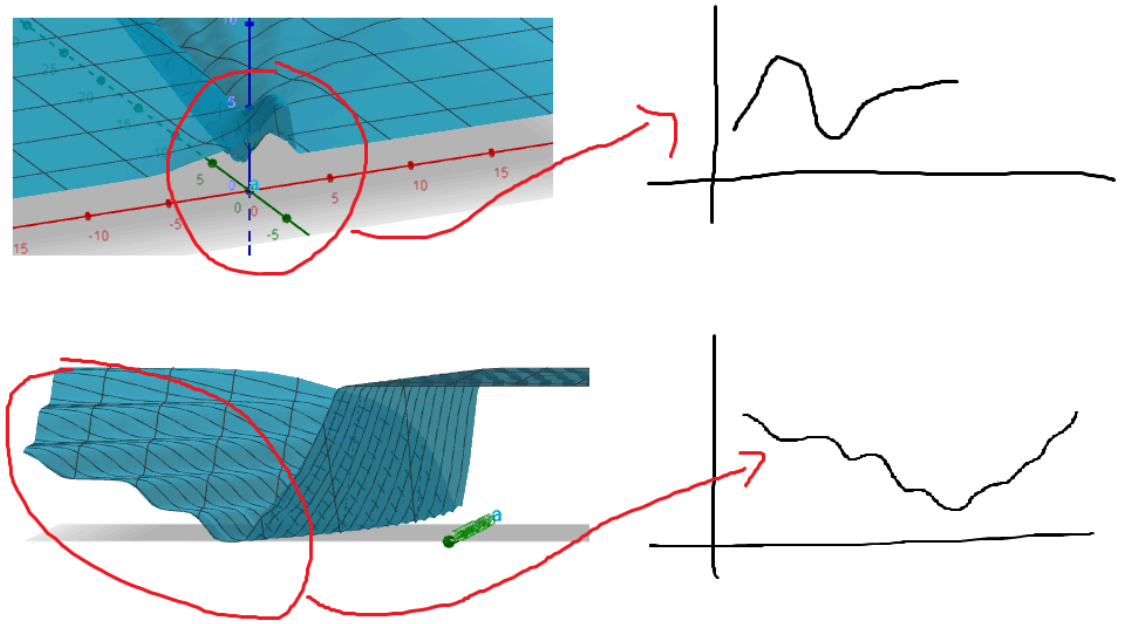


Simone Truglia

Enzo, **finora ho tenuto b a zero,** per evitare di avere il plot su due dimensioni, ma avendo solo la dimensione w.

Ho fatto una prova stamattina dello stesso plot in 3D con geogebra, aggiungendo anche theta2 (b) e si vede che inizia a uscire la struttura dei minimi locali quando ti muovi sull'asse b.

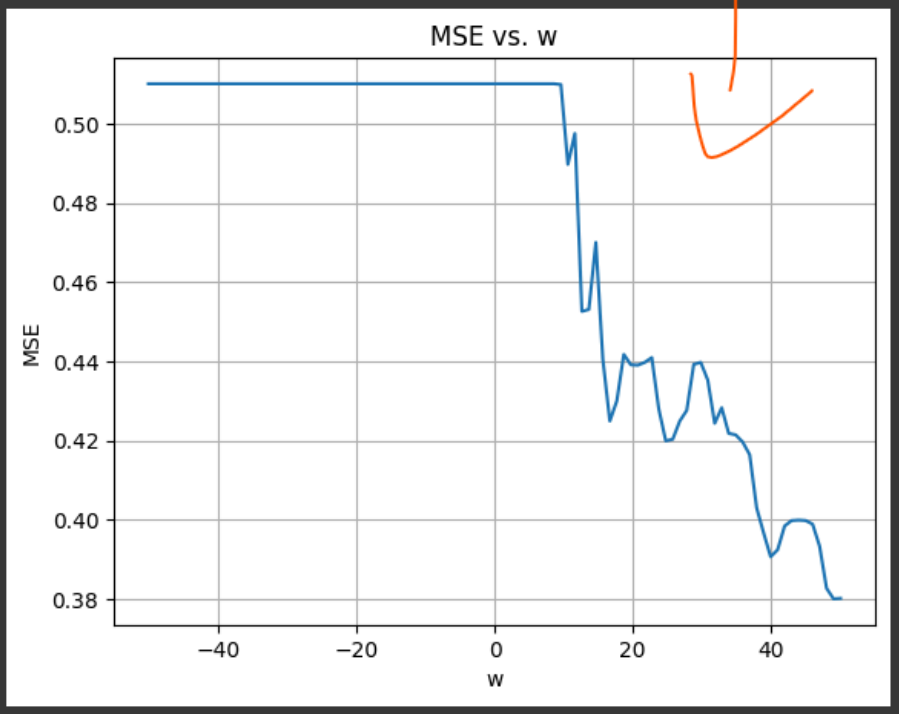
E' molto liscia su b vicino a zero (e sezionando b=0 da proprio la forma che veniva a me ieri) ma iniziano ad usicre le montagne russe appena ti sposti su b



Infatti, ho modificato il codice del notebook aggiungendo un b lontano da zero e inizia ad oscillare come un matto

```python
# Calculate the MSE for each value of w
mse_values = []
for w in w_values:
    y_pred = 1 / (1 + np.exp(-w * X + 100))
    mse = np.mean((y_pred - y) ** 2)
    mse_values.append(mse)

# Plot the MSE vs. w
plt.plot(w_values, mse_values)
plt.xlabel('w')
plt.ylabel('MSE')
plt.title('MSE vs. w')
plt.grid(True)
plt.show()
```



Grazie mille Enzo!

CONCLUSIONE

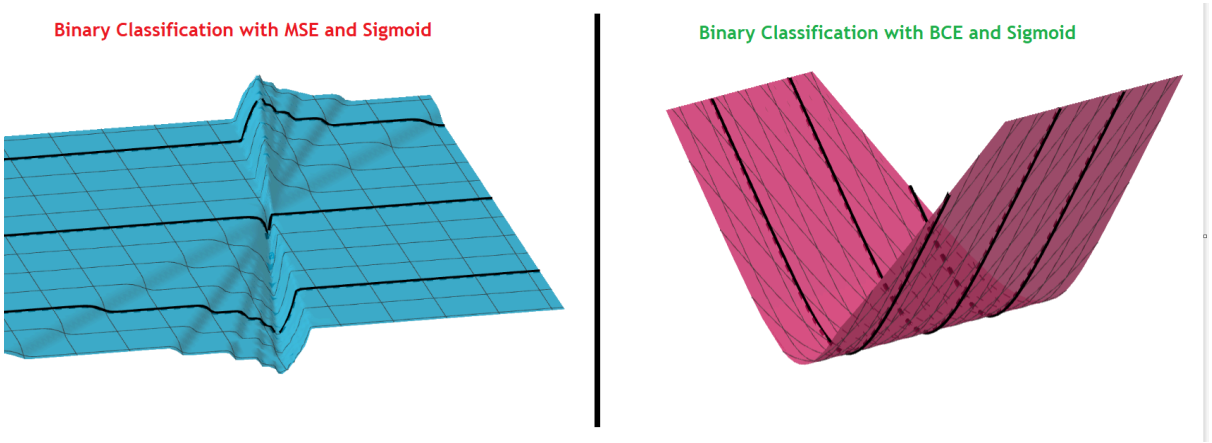
Ciao a tutti, dopo l'analisi che mi sono fatto tra ieri e oggi sul problema di classificazione binaria e la funzione di costo utilizzata, allego qui un'immagine carina da mettere negli appunti, sperando di fare cosa utile a qualcuno 🙂

In pratica, l'immagine mette in comparazione quello che succede, in due casi:

❌ Utilizzo della **MSE** con la **sigmoide** in un **problema di classificazione binaria** (sinistra)

✅ Utilizzo della **BCE** con la **sigmoide** in un **problema di classificazione binaria** (destra)

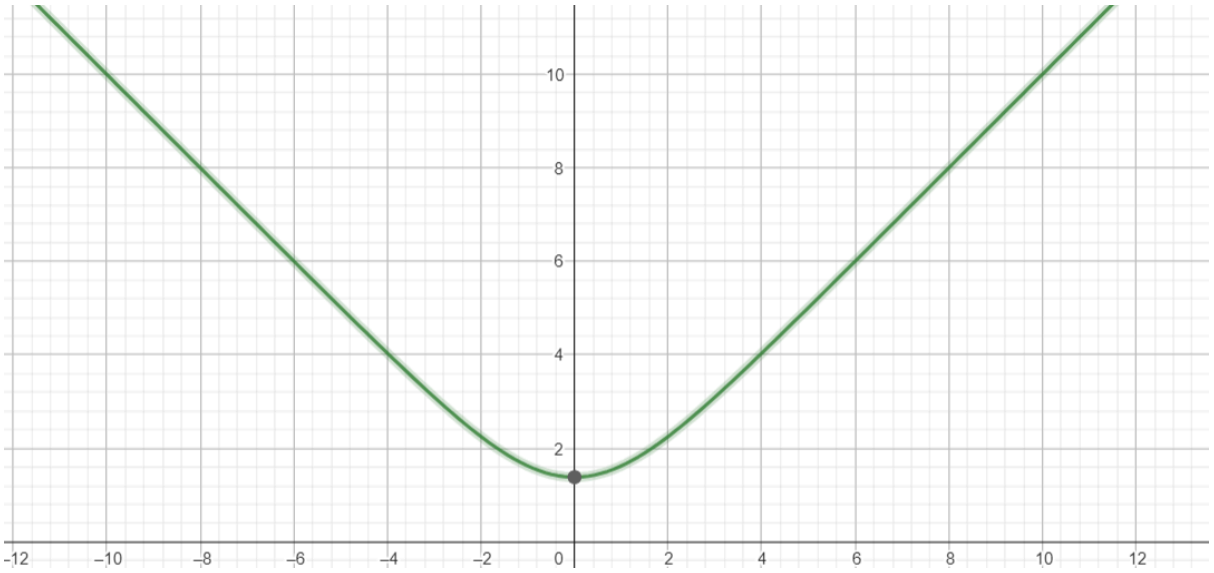
In entrambi i grafici, le singole funzioni sono in relazione ai soli due pesi w e b (quindi singola feature) e le linee di intersezione che vedete sono 3 linee su 3 valori di b fissati (b = 0 il centrale, una linea con b < 0 e una linea con b > 0), in modo da vedere a colpo d'occhio come si comporta la funzione solo su w



Dai grafici si vede bene perché la prima scelta è disastrosa: la probabilità di partire in un punto che ci porti ad un minimo globale è decisamente più bassa della probabilità di rimanere bloccati da qualche parte con il grandiente nullo, un po' per gli asintoti orizzontali sui lati, un po' per i vari minimi locali date dalle oscillazioni, che diventano via via sempre maggiori al crescere del valore assoluto di b (questo si vede bene guardando le intersezioni, non ci sono oscillazioni su b=0 ma diventano molto evidenti quando si allontana dall'origine). In più, al crescere del numero di paramentri in ballo (e quindi delle features), le oscillazioni diventano sempre più massicce e la probabilità di fermarsi in un minimo locale è molto alta.

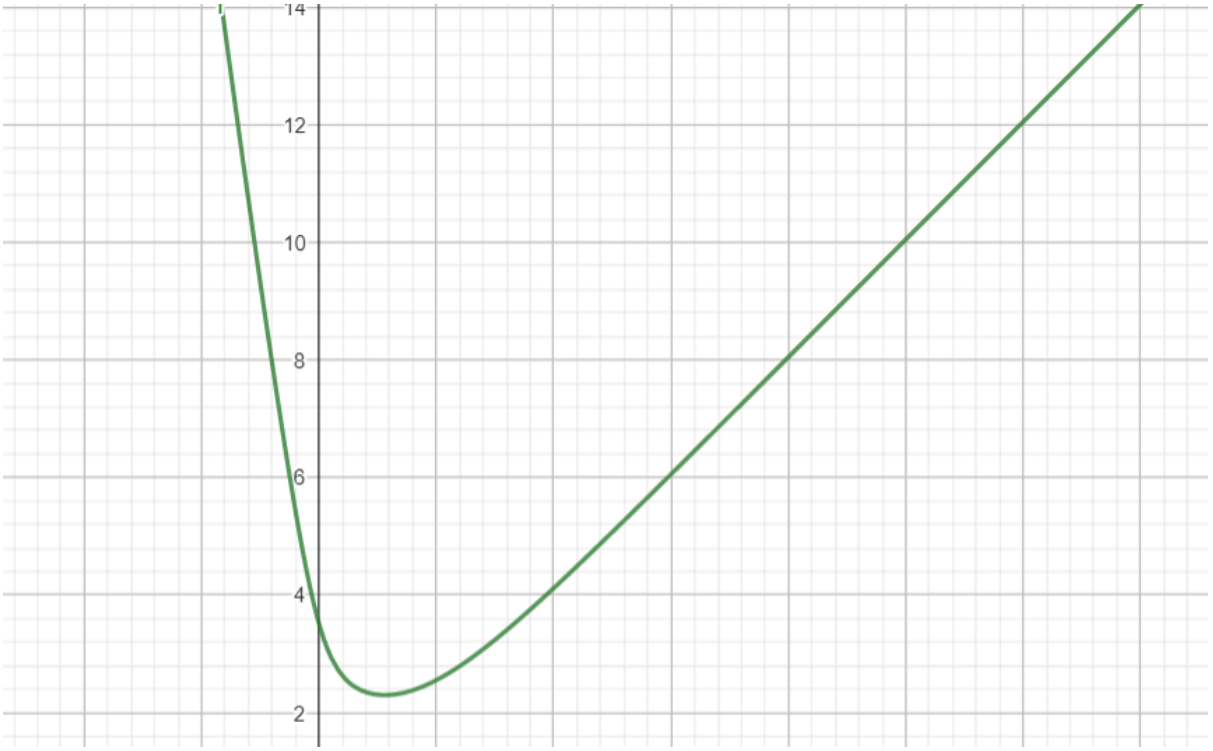
Allo stesso modo invece, a destra si vede come la seconda scelta è decisamente migliore per questa tipologia di problema

Occhio, anche se lo sembra dal plot in 3D, non è una parabola come nel caso della MSE nella regressione lineare, ma è la funzione data dalla somma dei logs (vedi screen sotto in 2d, che corrisponde ad una linea di intersezone con b=0)



nota anche che non è sempre così "precisa", ma dipende molto anche dai valori della sommatoria di X e Y.

Ad esempio, con altri valori a caso, l'intersezione di b = 0 verrebbe così.



Per me è stato un lavoro molto utile che mi ha aiutato a prendere confidenza con le funzioni di attivazioni / errore, in base alla situazione, spero vi sia utile 🙂