2. Elemente de bază ale rețelelor neuronale

2.1 Funcții de activare

Ieșirea unui nod al rețelei neuronale se obține prin calcularea sumei ponderate a intrărilor la care se adaugă valoarea de prag, urmată apoi de aplicarea unei funcții de activare (fig. 2.1).

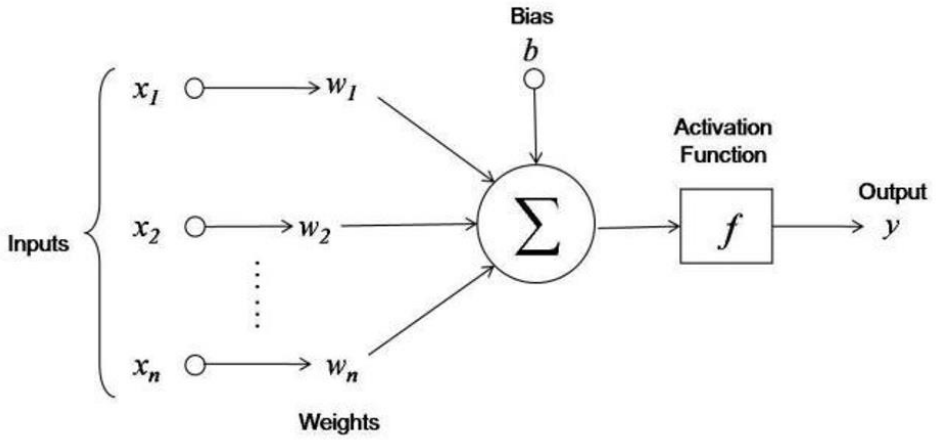

Fig. 2.1 Calculul ieşirii unui nod al rețelei neuronale (Grigoryan, 2022)

Funcția sigmoidă

Pentru ca algoritmul de propagare înapoi a erorii să funcționeze corect, autorii acestuia au înlocuit funcția de activare de tip treaptă cu funcția sigmoidă.

Funcția sigmoidă este definită ca:

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

Graficul funcției sigmoide este reprezentat în fig. 2.2.

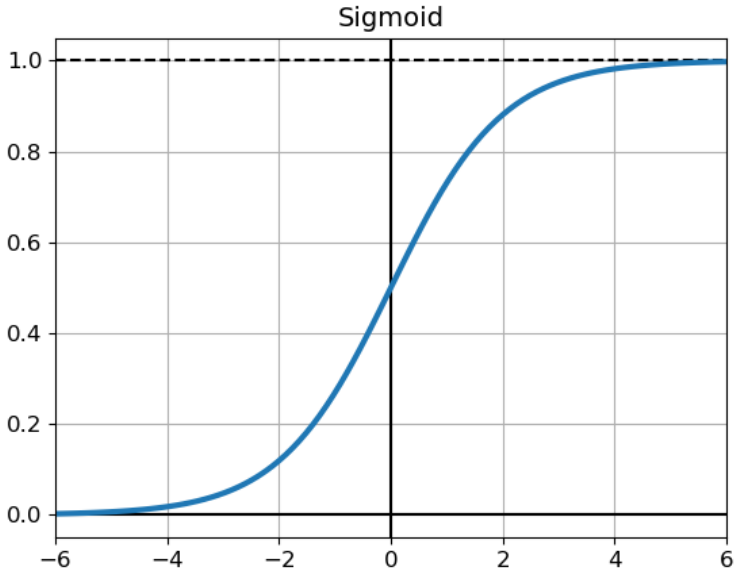

Fig. 2.2 Funcția de activare sigmoidă

Funcția tangentă hiperbolică

Funcția tangentă hiperbolică are o formă similară funcției sigmoide dar, spre deosebire de aceasta, ieșirea este centrată în jurul valorii 0.

Funcția tangentă hiperbolică este definită ca:

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

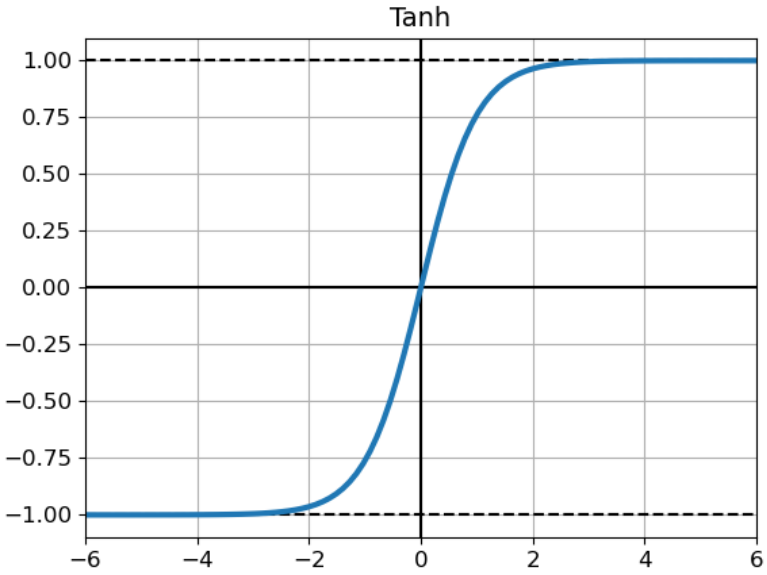

Fig. 2.3 Funcția de activare tangentă hiperbolică

Funcția ReLU (Rectified Linear Unit)

Funcția ReLU (Hahnloser et al., 2000) este continuă, dar nu este diferențiabilă în $z = 0$. Are însă avantajul de a fi ușor de calculat și funcționează foarte bine în practică.

Funcția ReLU este definită ca:

$$\mathrm{ReLU}(z) = \max(0, z)$$

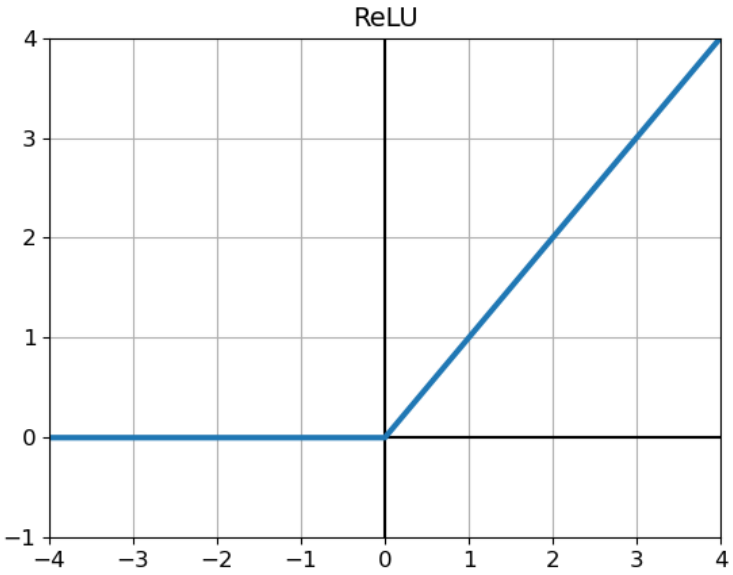

Fig. 2.4 Funcția de activare ReLU

Funcția softplus

Funcția softplus este o aproximare (o variantă netedă) a funcției ReLU. Ca și în cazul ReLU, funcția softplus are întotdeauna valori pozitive la ieșire.

Funcția softplus este definită ca:

$$\text{softplus}(z) = \ln\left(e^{z} + 1\right)$$

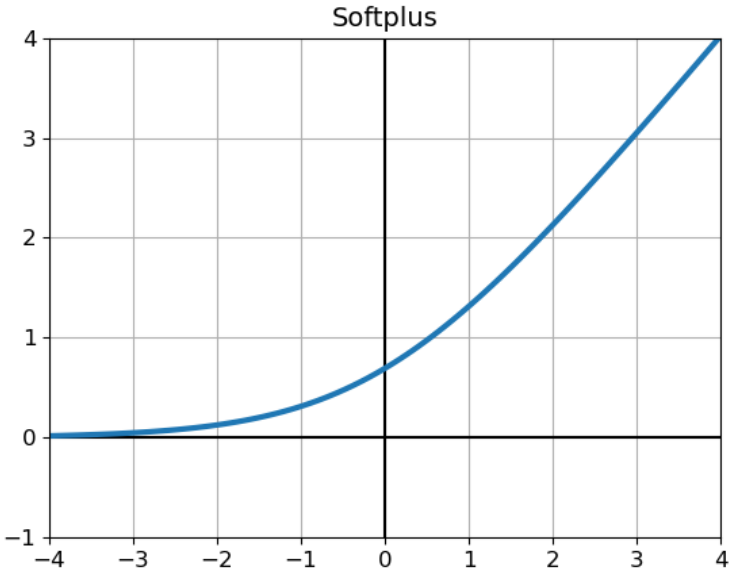

Fig. 2.5 Funcția de activare softplus

Funcția leaky ReLU

Funcția ReLU nu se saturează pentru valori pozitive, însă derivata acesteia este 0 atunci când intrarea este negativă. Funcția leaky ReLU (Maas et al., 2013) rezolvă această situație, fiind definită ca:

$$\text{LeakyReLU}(z) = \max(\alpha z, z),$$

unde α este un hiperparametru și reprezintă panta funcției pentru $z < 0$.

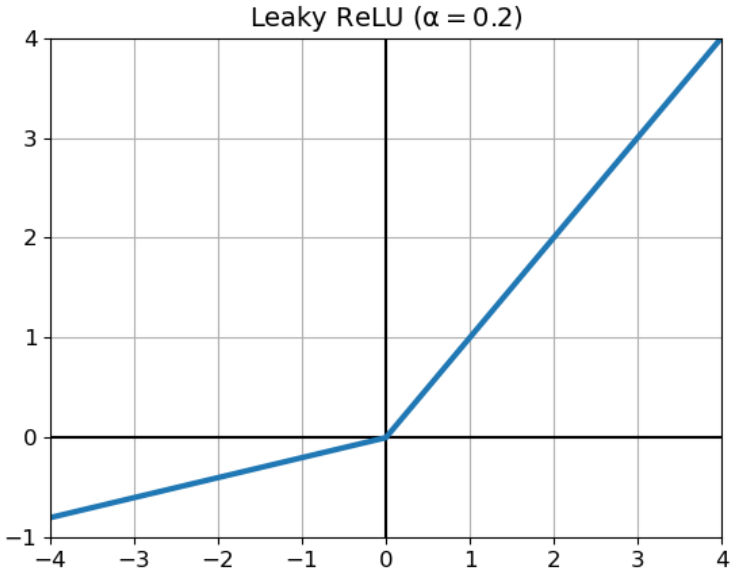

Fig. 2.6 Funcția de activare leaky ReLU

Parametric leaky ReLU (PReLU) este o variantă propusă de He et al. (2015), în care α, de data aceasta, este un parametru învățat în timpul antrenării prin propagare înapoi.

Randomized leaky ReLU (RReLU) este o altă variantă propusă și utilizată în cadrul competiției Kaggle NDSB (Kaggle National Data Science Bowl Competition[1]), în care α este o valoare aleatorie cu distribuție uniformă într-un interval dat în timpul antrenării, iar în timpul testării este aleasă valoarea medie din timpul antrenării.

Funcția ELU

Exponential linear unit (ELU) este o funcție de activare propusă de Clevert et al. (2015), fiind definită ca:

$$\mathrm{ELU}(z) = \begin{cases} \alpha\left(e^{z} - 1\right) & \text{dacă } z < 0 \\ z & \text{dacă } z \geq 0 \end{cases}$$

Hiperparametrul α este, de obicei, setat la 1, iar dreapta $y = -\alpha$ este o asimptotă orizontală negativă.

[1] https://www.kaggle.com/c/datasciencebowl

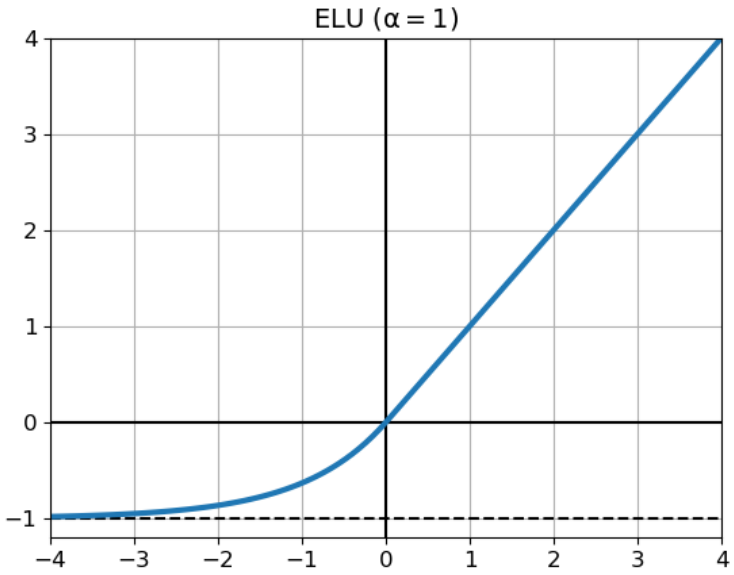

Fig. 2.7 Funcția de activare ELU

Funcția SELU

Scaled ELU (SELU), propusă de Klambauer et al. (2017), este o variantă scalată a funcției de activare ELU.

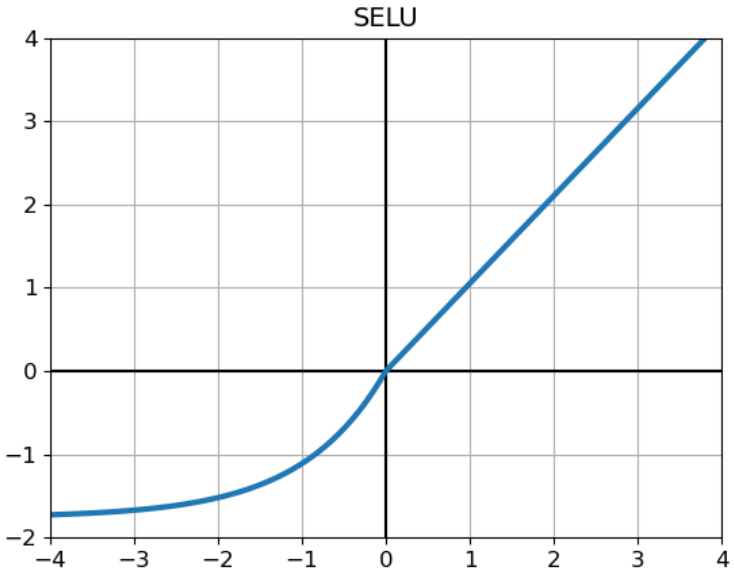

Fig. 2.8 Funcția de activare SELU

Funcția softmax

Funcția softmax (numită și exponențiala normalizată) este definită ca:

$$\operatorname{softmax}(z_i) = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}},$$

unde z_i este suma ponderată a intrărilor nodului de ieșire *i*, iar *K* este numărul nodurilor stratului de ieșire.

Valorile la ieșirea funcției softmax sunt cuprinse între 0 și 1 iar suma acestora este 1. Astfel, funcția de activare softmax este o alegere potrivită în cazul clasificării multiclasă.

Exemplu de calcul:

```
import tensorflow as tf

inputs = tf.constant([[4, 2, -2]], dtype=tf.float32)
outputs = tf.keras.activations.softmax(inputs)

print(outputs.numpy())
print(tf.reduce_sum(outputs).numpy())
```

```
[[0.87887824 0.11894324 0.00217852]]
1.0
```

2.2 Funcții de cost

Funcția de cost măsoară eroarea rețelei neuronale, adică diferența dintre ieșirea acesteia și valorile țintă.

2.2.1 Funcții de cost folosite în cazul regresiei

***Eroarea pătratică medie* (*MSE*)**

$$MSE = \frac{\sum_{i=1}^{N} (y_i - \hat{y}_i)^2}{N},$$

unde y_i este valoarea țintă, $\hat{y}_i$ este predicția rețelei neuronale, iar N este numărul de instanțe dintr-un lot.

***Eroarea absolută medie* (*MAE*)**

$$MAE = \frac{\sum_{i=1}^{N} |y_i - \hat{y}_i|}{N}.$$

În fig. 2.9 este prezentată o comparație între *MSE* și *MAE*.

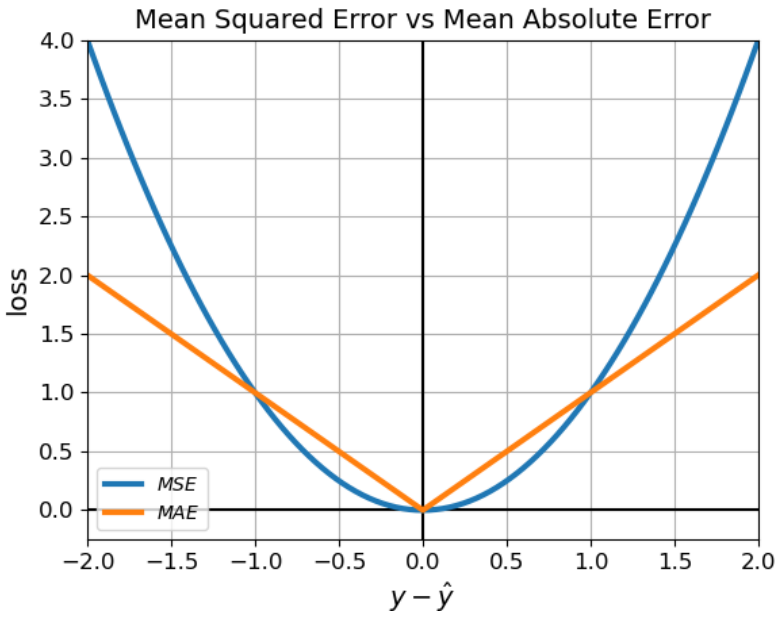

Fig. 2.9 Eroarea pătratică medie și eroarea absolută medie

Funcția de cost Huber

Funcția de cost utilizată în timpul antrenării în cazul regresiei este, de obicei, eroarea pătratică medie, dar dacă există multe valori aberante în setul de antrenare, este de preferat să se utilizeze eroarea absolută medie.

Alternativ, se poate utiliza funcția de cost Huber, care este o combinație a celor două.

Funcția de cost Huber este definită ca (Hastie et al., 2009):

$$\textit{Huber loss} = \begin{cases} \frac{1}{2}\left(y_i - \hat{y}_i\right)^2 & \text{for } \left|y_i - \hat{y}_i\right| \le \delta, \\ \delta\left(\left|y_i - \hat{y}_i\right| - \frac{1}{2}\delta\right), & \text{otherwise.} \end{cases}$$

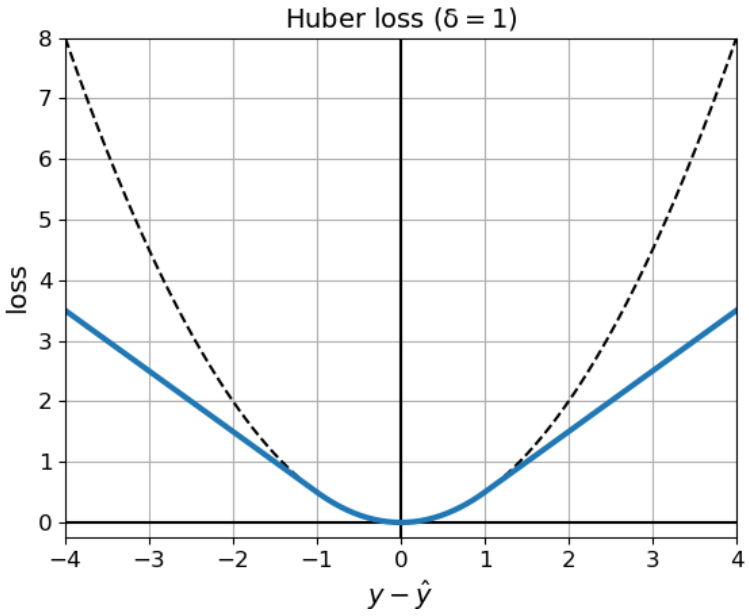

Fig. 2.10 Funcția de cost Huber pentru $\delta = 1$

2.2.2 Funcții de cost folosite în cazul clasificării

Entropia încrucișată (Cross-Entropy)

Entropia încrucișată (*CE*) și eroarea pătratică medie (*MSE*) sunt funcții de cost utilizate pe scară largă pentru antrenarea modelelor de învățare automată.

În timp ce *MSE* este utilizată, de obicei, pentru modelele de regresie liniară, *CE* este o opțiune prioritară pentru modelele de clasificare.

Formula de calcul pentru entropia încrucișată este:

$$CE = -\frac{\sum_{i=1}^{N}\sum_{j=1}^{C} y_{i,j} \ln \hat{y}_{i,j}}{N},$$

unde $y_{i,j}$ este valoarea așteptată, $\hat{y}_{i,j}$ este valoarea prezisă, *C* este numărul de clase, iar *N* este numărul de instanțe dintr-un lot.

În fig. 2.11 este prezentată o comparație între *CE* și *MSE*.

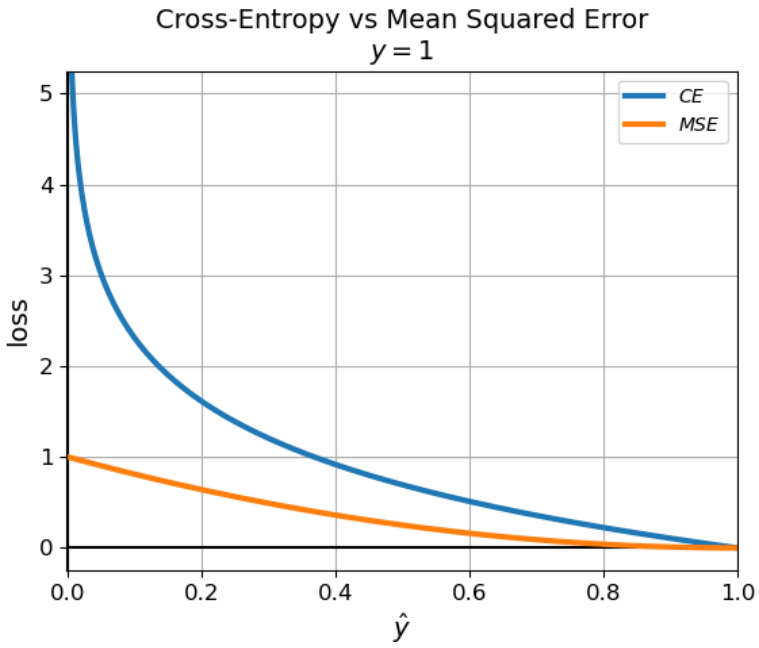

Fig. 2.11 Diferența dintre entropia încrucișată (CE) și eroarea pătratică medie (MSE)

Funcția de cost utilizată în cazul modelelor de clasificare binară este *entropia încrucișată binară* (*BCE*), în timp ce pentru modelele de clasificare multiclasă se folosește ca funcție de cost *entropia încrucișată categorială* (*CCE*).

Formula de calcul pentru entropia încrucișată binară este:

$$BCE = -\frac{\sum_{i=1}^{N}\left[y_i \ln \hat{y}_i + (1-y_i)\ln(1-\hat{y}_i)\right]}{N},$$

unde y_i este valoarea așteptată, $\hat{y}_i$ este valoarea prezisă, iar N este numărul de instanțe dintr-un lot.

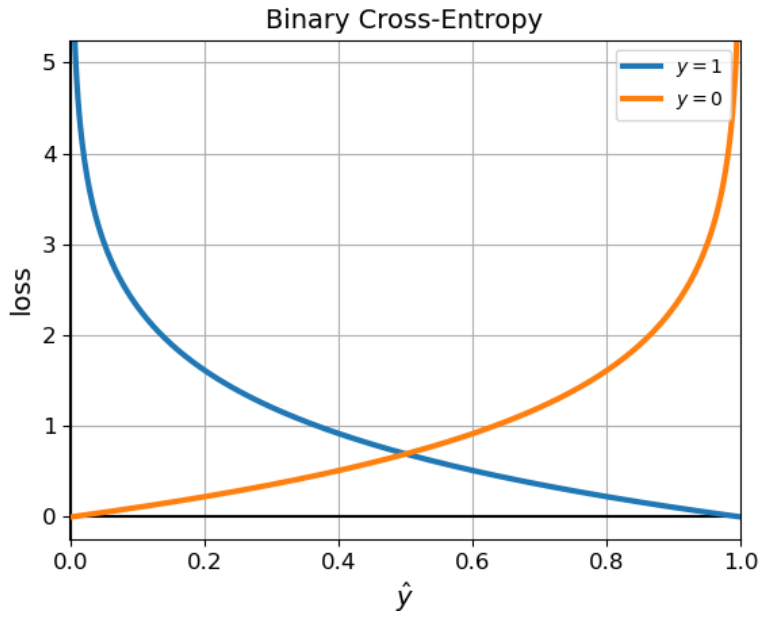

Fig. 2.12 Entropia încrucișată binară (BCE) în funcție de valoarea prezisă ($\hat{y}$)

Exemple de calcul:

```
import tensorflow as tf

y_true = [[0, 1, 0], [0, 0, 1]]
y_pred = [[0.05, 0.95, 0], [0.1, 0.8, 0.1]]

cce = tf.keras.losses.CategoricalCrossentropy(
    reduction='sum_over_batch_size')
print(cce(y_true, y_pred).numpy())

cce = tf.keras.losses.CategoricalCrossentropy(
    reduction='sum')
print(cce(y_true, y_pred).numpy())

cce = tf.keras.losses.CategoricalCrossentropy(
    reduction='none')
print(cce(y_true, y_pred).numpy())
```

```
1.1769392
2.3538785
[0.05129331 2.3025851 ]
```

```
y_true = [1, 2]
y_pred = [[0.05, 0.95, 0], [0.1, 0.8, 0.1]]

scce = tf.keras.losses.SparseCategoricalCrossentropy()
print(scce(y_true, tf.convert_to_tensor(y_pred)).numpy())
```

```
1.1769392
```

```
y_true = [0, 1, 0, 0]
y_pred = [0.6, 0.3, 0.2, 0.8]

bce = tf.keras.losses.BinaryCrossentropy()
print(bce(y_true, y_pred).numpy())

bce = tf.keras.losses.BinaryCrossentropy(reduction='none')
print(bce(y_true, y_pred).numpy())
```

```
0.988211
[0.9162906  1.2039725  0.22314338 1.6094375 ]
```

2.3 Utilizări ale rețelelor neuronale

2.3.1 Regresia

În cazul regresiei, dacă dorim să prezicem o singură valoare (de exemplu, prețul unei case, având în vedere mai multe caracteristici ale acesteia), atunci avem nevoie doar de un singur nod pe stratul de ieșire.

Dacă însă dorim să prezicem mai multe valori simultan (*regresie multivariată*), atunci avem nevoie de câte un nod de ieșire pentru fiecare valoare prezisă.

Tabelul 2.1 Arhitectura tipică a unei rețele neuronale în cazul regresiei (Géron, 2022)

Hiperparametru	**Valoare tipică**
Numărul nodurilor de intrare	Câte unul pentru fiecare caracteristică de intrare
Numărul straturilor ascunse	Depinde de problemă (de obicei, între 1 și 5)
Numărul nodurilor per strat ascuns	Depinde de problemă (de obicei, între 10 și 100)
Numărul nodurilor de ieșire	Câte unul pentru fiecare valoare prezisă
Funcția de activare în straturile ascunse	ReLU sau SELU
Funcția de activare în stratul de ieșire	Niciuna, ReLU/softplus (dacă valorile de ieșire sunt pozitive), sau sigmoid/tanh (dacă valorile de ieșire se încadrează într-un anumit interval)
Funcția de cost	MSE sau MAE/Huber (dacă există valori anormale)

2.3.2 Clasificarea

Tabelul 2.2 Arhitectura tipică a unei rețele neuronale în cazul clasificării (Géron, 2022)

Hiperparametru	**Clasificare binară**	**Clasificare binară multietichetă**	**Clasificare multiclasă**
Stratul de intrare și straturile ascunse	La fel ca în cazul regresiei	La fel ca în cazul regresiei	La fel ca în cazul regresiei
Numărul nodurilor de ieșire	Unul	Câte unul pentru fiecare etichetă	Câte unul pentru fiecare clasă
Funcția de activare în stratul de ieșire	Sigmoid	Sigmoid	Softmax
Funcția de cost	Entropia încrucișată	Entropia încrucișată	Entropia încrucișată