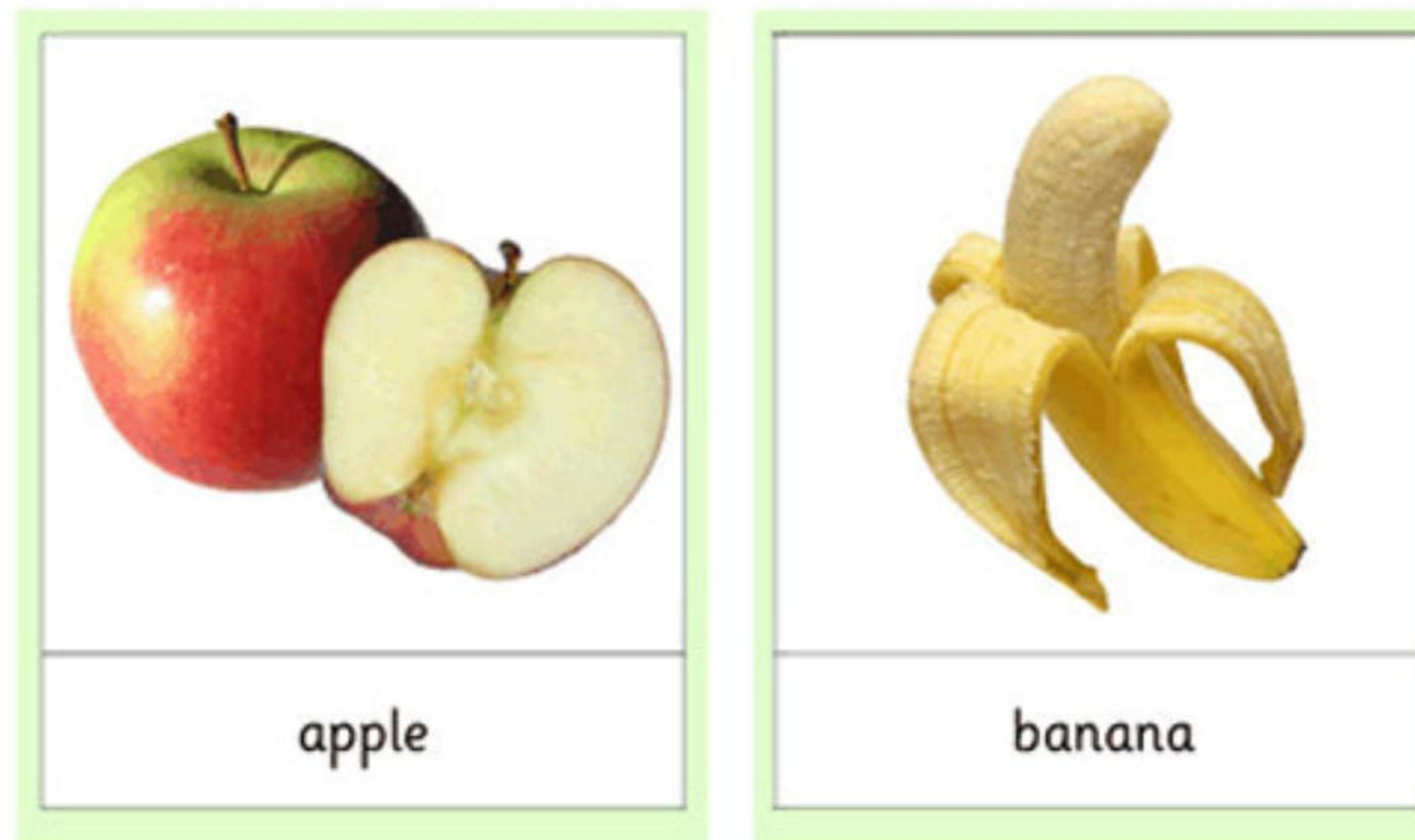


# Multi-Class Classification

## Lecture 7 - Part 1

شرح المفاهيم الأساسية وتطبيقاتها في الـ Neural Networks

# ما هي Multi-Class Classification



- عندما يكون لدينا أكثر من **Class** واحد ممكن للتصنيف.
- أمثلة عملية:
  - تصنيف الفاكهة:  
**{apple, banana, orange}**
  - تصنيف الملابس:  
**{t-shirt, jeans, dress}**
- في الوضع الطبيعي، كل عينة تتبع لـ **Class** واحد فقط، وهذا ما يسمى بـ **Single-Label**.

# | الفرق بين Soft Label و Hard Label |

## Soft Label

يتم تمثيله ك **Distribution** (توزيع احتمالات).

$$y = [0.1, 0.7, 0.2]$$

يعطي درجة ثقة لكل كلاس. مفيد في  
و **Label Smoothing**  
. **Knowledge Distillation**

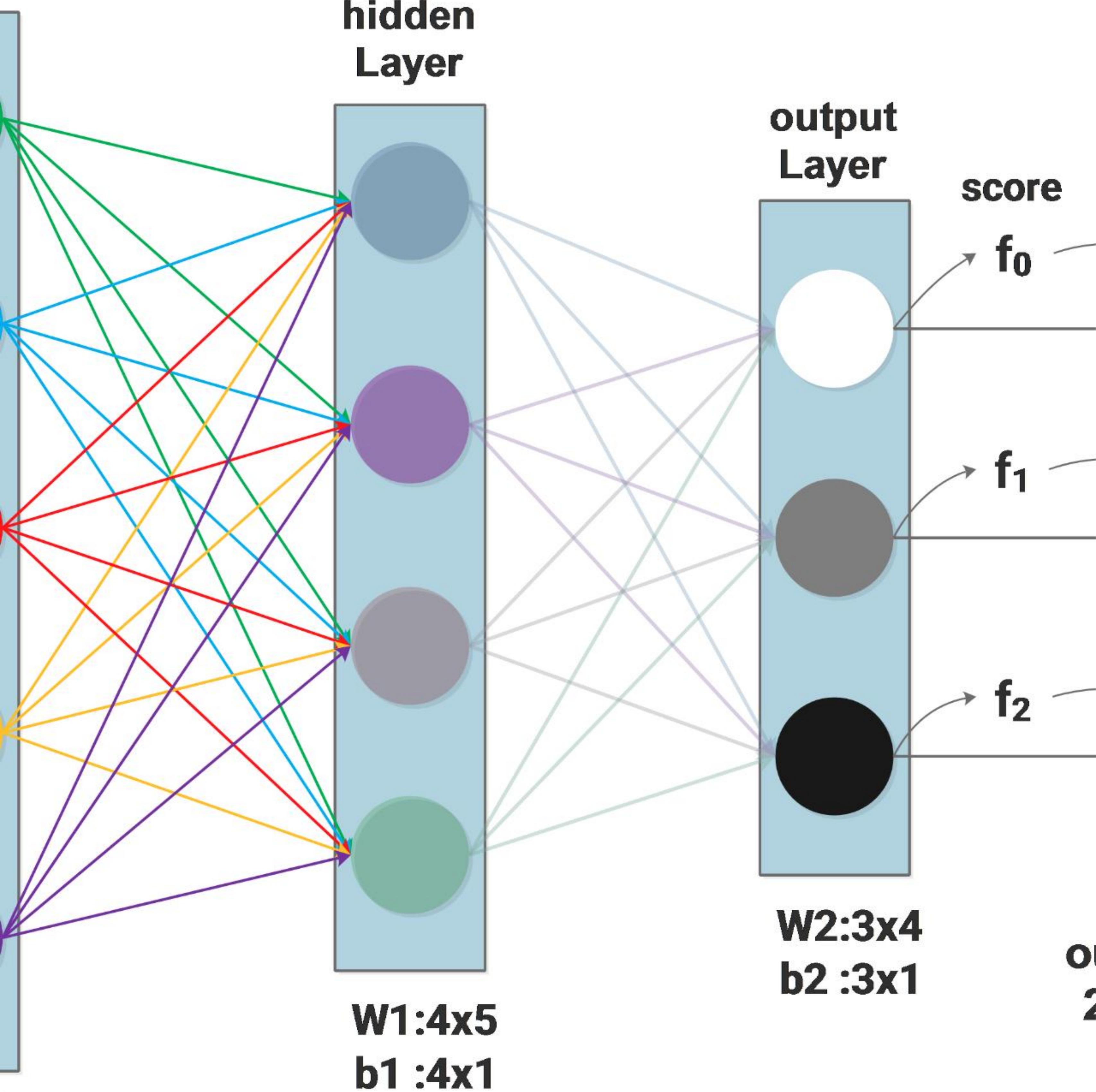
## Hard Label

يتم تمثيله ك **One-Hot Vector**.

$$y = [0, 1, 0]$$

الرقم 1 للكلاس الصحيح والباقي أصفار. يستخدم غالباً في **Supervised Learning**.

# دالة Softmax



تقوم بتحويل الـ **Vector of Logits (z)** إلى احتمالات حقيقة.

$$p_k = \frac{e^{z_k}}{\sum_{j=1}^K e^{z_j}}$$

- المخرجات دائمًا موجبة.
- مجموع الاحتمالات يساوي 1.
- الدالة الأEXPONENTIAL تضخم الفروقات بين القيم.

# | دالة الخسارة | Categorical Cross-Entropy

الشكل العام

تستخدم مع **Softmax** لقياس الخطأ:

$$L = - \sum_{k=1}^K y_k \cdot \log(p_k)$$

في حالة **Hard Label**

إذا كان الكلاس الصحيح هو  $c$ , فإن المعادلة تبسط إلى:

$$L = - \log(p_c)$$

- إذا كان النموذج متأكداً ( $p \approx 1$ ) ← الخسارة ≈ 0.
- إذا كان مخطئاً ( $p \approx 0$ ) ← الخسارة كبيرة جدًا.

حيث  $y$  هو التوزيع الحقيقي و  $p$  هو الاحتمال المتوقع.

# | طرق حساب الـ Error |



## Evaluation Metric

### Accuracy

نسبة التوقعات الصحيحة من إجمالي العينات. تُستخدم لتقدير أداء النموذج نهائياً.

$$\text{Error Rate} = 1 - \text{Accuracy}$$



## Training Loss

### Cross-Entropy Loss

تُستخدم لتدريب النموذج وحساب الـ Gradients لتحديث الأوزان (Weights).

# Logits vs Softmax Probabilities | مقارنة :

توضيح كيف تقوم Softmax بتحويل القيم (Logits) إلى احتمالات مجموعها 1.



للحظ أن مجموع الاحتمالات (الجانب الأيسر) يساوي 1 (%100)

## Input a set of training examples

For each training example  $x$ : Set the corresponding input activation  $a^{x,1}$ , and perform the following steps:

- Feedforward: For each  $l = 2, 3, \dots, L$  compute

$$z^{x,l} = w^l a^{x,l-1} + b^l \text{ and } a^{x,l} = \sigma(z^{x,l}).$$

- Output error  $\delta^{x,L}$ : Compute the vector

$$\delta^{x,L} = \nabla_a C_x \odot \sigma'(z^{x,L}).$$

- Backpropagate the error: For each

$l = L - 1, L - 2, \dots, 2$  compute

$$\delta^{x,l} = ((w^{l+1})^T \delta^{x,l+1}) \odot \sigma'(z^{x,l}).$$

Gradient descent: For each  $l = L, L - 1, \dots, 2$  update the weights according to the rule  $w^l \rightarrow w^l - \frac{\eta}{m} \sum_x \delta^{x,l} (a^{x,l-1})^T$ , the biases according to the rule  $b^l \rightarrow b^l - \frac{\eta}{m} \sum_x \delta^{x,l}$ .

## تحديث الأوزان | Weight Update

الـ Gradient بالنسبة لـ Logits هو الفرق بين التوقع والحقيقة:

$$\frac{\partial L}{\partial z} = p - y$$

• إذا كان  $y > p$ : النموذج بالغ في الاحتمال ← نقل الوزن.

• إذا كان  $y < p$ : النموذج قلل الاحتمال ← نزيد الوزن.

يتم استخدام هذا المبدأ في

Gradient Descent

```
3  
4 #this say function is the most important part of kids programming  
5 #it uses the built in OSX say command to convert text to speech  
6 def say(something):  
7     system('say "{}"'.format(something))  
8  
9     #how big is the number?  
10    max_number = 100  
11    first_line = "The number is between 0 and 100."  
12    print(first_line)  
13    say(first_line)  
14    number = int(input("What's the number?"))  
15    not_so_low = number >= 10  
16    not_so_high = number <= 90  
17  
18    #keep asking until they get it right  
19    while not_so_low and not_so_high:  
20        answer = input("Is it higher or lower?")  
21        you_said = int(answer)  
22        say(you_said)  
23        if answer == "higher":  
24            say("I'm thinking of a number between 10 and 90.")  
25            say("Is it higher or lower?")  
26            say("The number is higher.")  
27            say("Is it higher or lower?")  
28            say("The number is higher.")
```

# تطبيقات لمهندسي الذكاء الاصطناعي

**تمثيل الصور الطبيعية، المركبات، إلخ.**: **Image Classification** ✓

**فهم نية المستخدم في الشات بوت.**: **NLP Intent Detection** ✓

**استخدام نماذج Soft Labels لتدريب**: **Knowledge Distillation** ✓

أصغر وأذع.

؟äñîñä

شکرآ لاستھنائے کم

Lecture 7 - Part 1: Multi-Class Classification

# Image Sources |

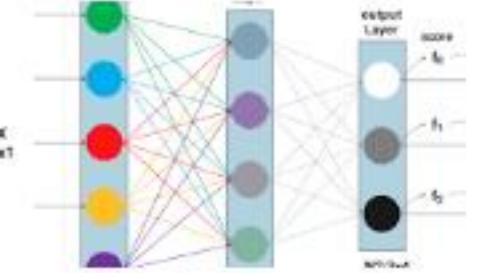
[http://absorbentminds.co.uk/cdn/shop/products/xl\\_15\\_121.jpg?v=1667820307](http://absorbentminds.co.uk/cdn/shop/products/xl_15_121.jpg?v=1667820307)

Source: [absorbentminds.co.uk](http://absorbentminds.co.uk)



[https://www.wangxinliu.com/images/machine\\_learning/2layer\\_nn.png](https://www.wangxinliu.com/images/machine_learning/2layer_nn.png)

Source: [www.wangxinliu.com](https://www.wangxinliu.com)



<https://i.sstatic.net/goLCZ.png>

Source: [cs.stackexchange.com](https://cs.stackexchange.com)

For each training example: (1) Set the corresponding input activation  $a^{(0)}$ , and perform the following steps:  
o Feedforward: For each  $i = 1, \dots, L$ , compute  
 $\hat{z}^{(i)} = \sigma(a^{(i)}) = \sigma^i$  and  $a^{(i+1)} = \sigma(\hat{z}^{(i)})$ .  
o Output error: Compute the vector  
 $\delta^{(L)} = \nabla_{\theta^{(L)}} \mathcal{L}(y, \hat{y})$ .  
o Backpropagate the error: For each  
 $i = L-1, \dots, 1$ , compute  
 $\delta^{(i)} = (\sigma'(\hat{z}^{(i)}))^\top \delta^{(i+1)}$ .

-<https://cdn-media-1.freecodecamp.org/images/h5iAZ9X9kElc-pY9HEnd8jrPjSpDWoJQJeK>

Source: [www.freecodecamp.org](https://www.freecodecamp.org)

