

Ứng dụng Trí tuệ nhân tạo trong Nuôi trồng thủy sản

NGUYỄN HẢI TRIỀU¹

¹ Bộ môn Kỹ thuật phần mềm,
Khoa Công nghệ thông tin, Trường ĐH Nha Trang

NhaTrang, September 2024

1 Training Multilayer Neural Networks

- Logistic Regression for Multiple Classes

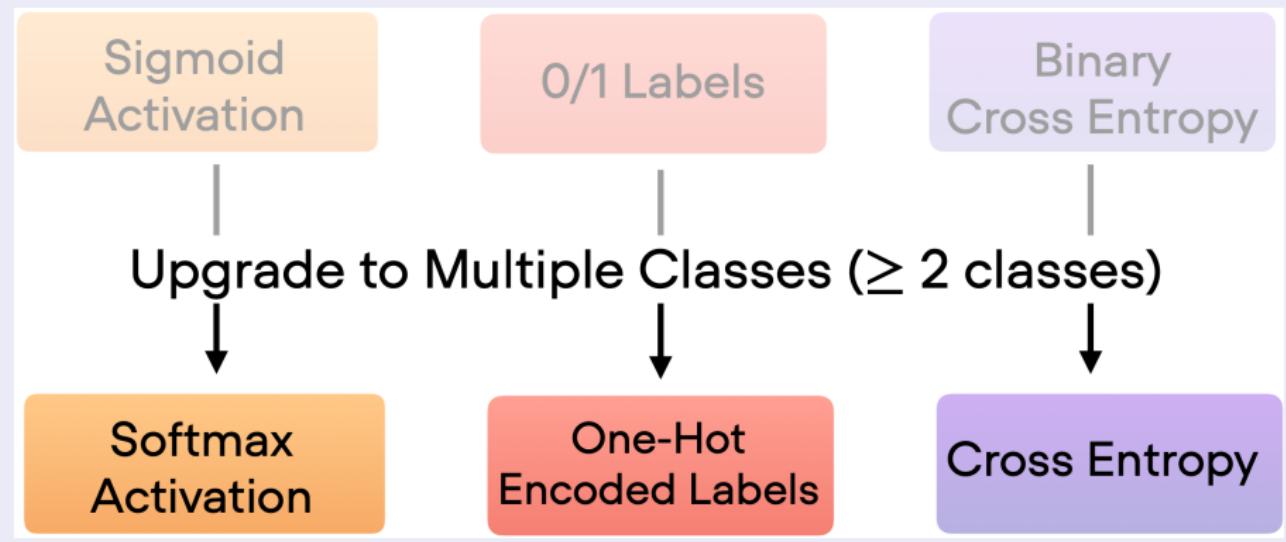
Huấn luyện mạng Neural Network nhiều lớp

Trong phần trước, ta đã đề cập đến bài toán phân loại nhị phân Logistic Regression. Trong phần này, chúng ta sẽ làm việc với bài toán phân loại có nhiều hơn 2 lớp (*multiple classes*).

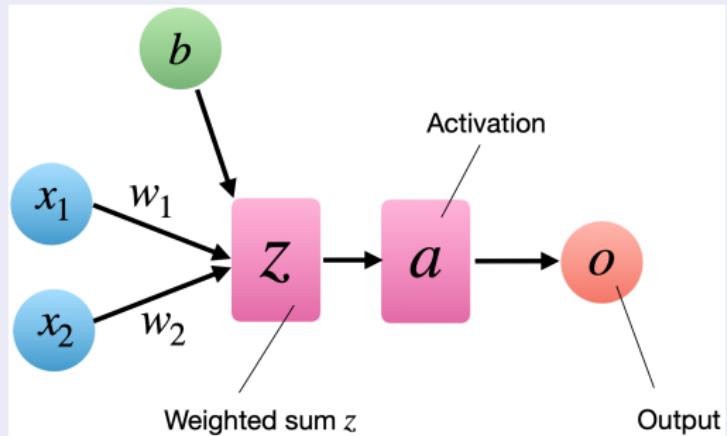
From logistic regression to multilayer neural networks

From logistic regression to multilayer neural networks is to *modify the logistic regression model so that it can deal with multiple classes*. The multi-class logistic regression is commonly called softmax regression.

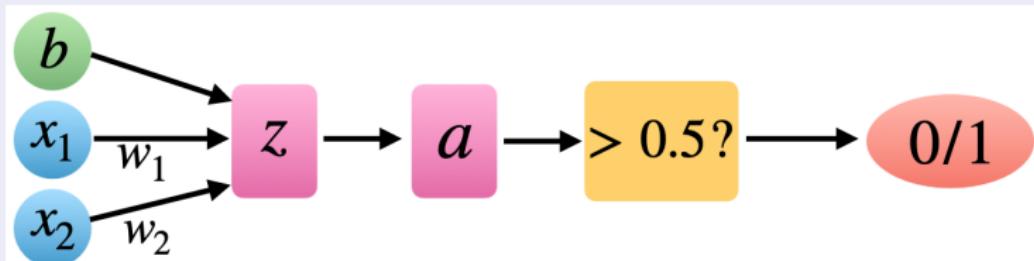
Previous Binary Classification Setting (2 Classes)



General Structure of Single Layer Neural Nets

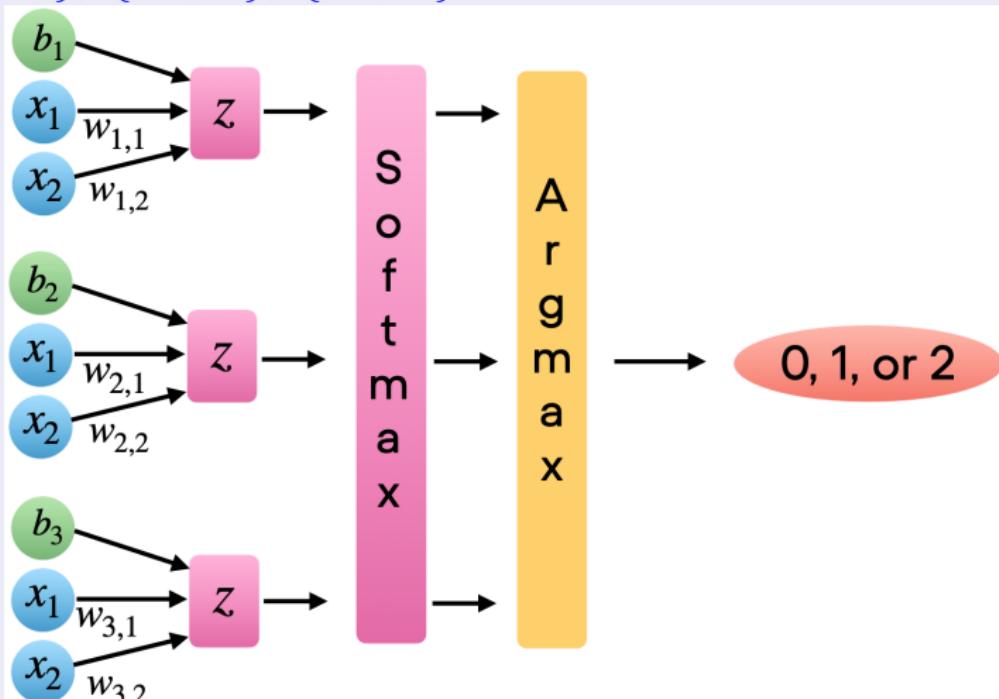


Binary Logistic Regression Model

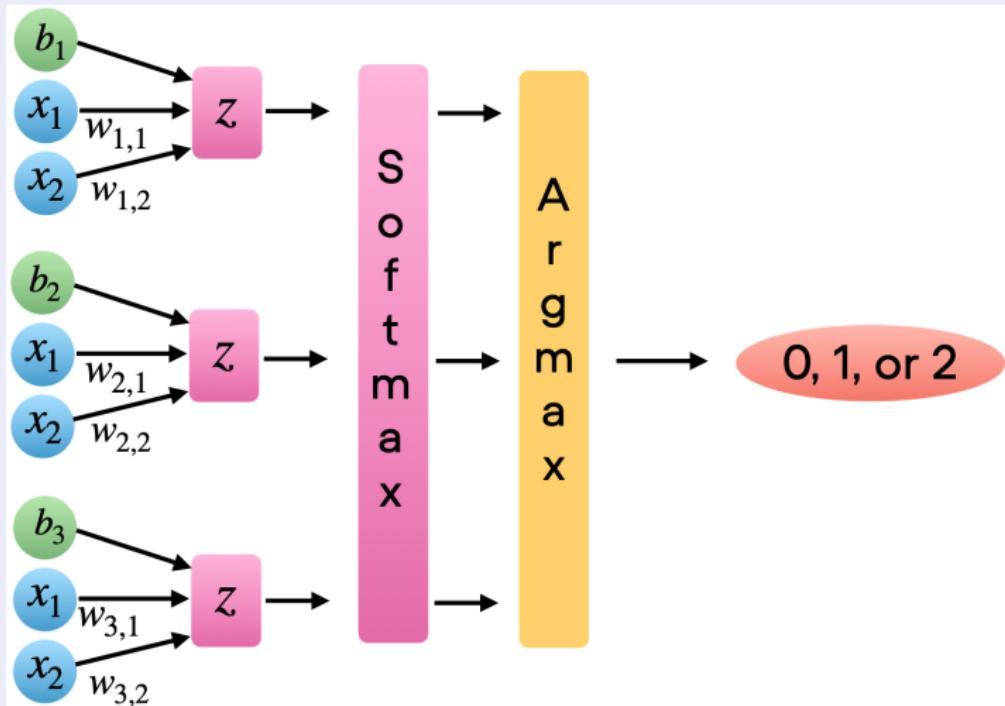


Binary → Softmax Regression

Tương tự như mô hình phân loại nhị phân, ta xem xét ví dụ bài toán phân loại 3 lớp 0, 1, 2. Đầu tiên chúng ta cần có 3 tập trọng số $\{\mathbf{w}_1, b_1\}$, $\{\mathbf{w}_2, b_2\}$, $\{\mathbf{w}_3, b_3\}$.

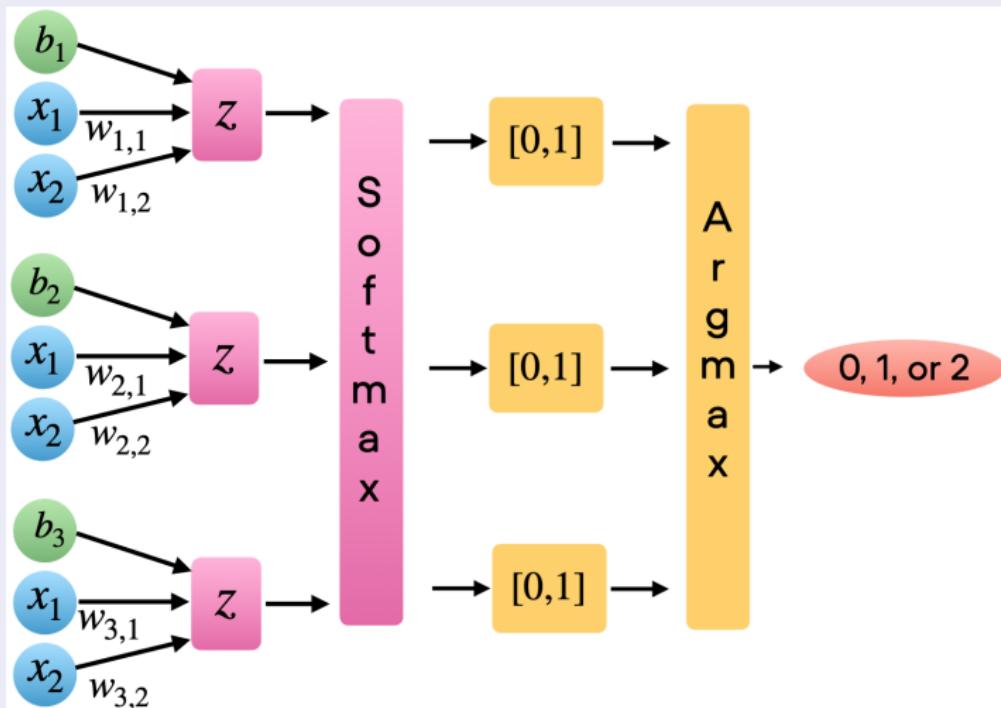


Binary → Softmax Regression



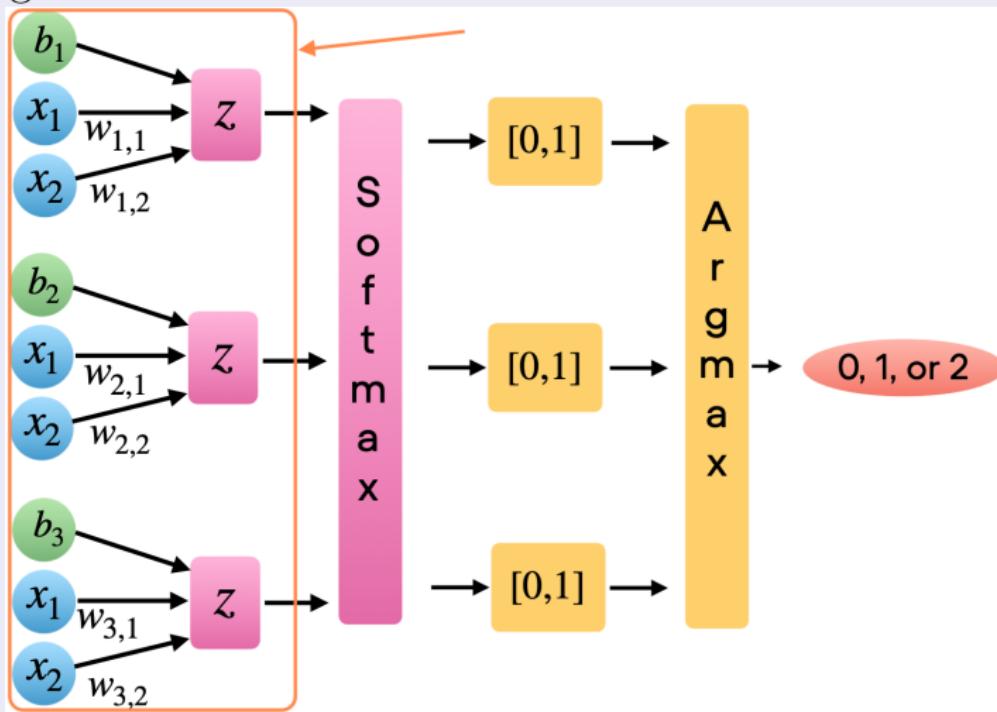
Activation a từ sigmoid được thay bằng **Softmax**, hàm threshold từ > 0.5 được thay bằng **Argmax**.

Binary → Softmax Regression



Output của Softmax cũng trả ra các giá trị 0, 1 như sigmoid.

Đầu tiên, ta tập trung vào phần tính toán weighted sum z với 3 bộ trọng số



Dot Product Plus Bias Unit → Matrix Multiplication Plus Bias Vector

x: Single training example

```
import torch

x = torch.tensor([1.1, 2.1])
w = torch.tensor([0.1, 0.2])
b = torch.tensor(0.5)
```

```
torch.matmul(x, w) + b
```

tensor(1.0300)

(a)

```
x = torch.tensor([[1.1, 2.1]])

w = torch.tensor([[0.1, 0.2],
                 [0.5, 0.6],
                 [0.8, 0.9]])
b = torch.tensor([0.5, 0.6, 0.7])

torch.matmul(x, w.T) + b
```

tensor([[1.0300, 2.4100, 3.4700]])

(b)

Hình 1: Ví dụ cho trường hợp 3 classes, ta bổ sung thêm 3 bộ trọng số.

Model Code Changes

```
logreg.py > LogisticRegression
1 import torch
2
3 class SoftmaxRegression(torch.nn.Module):
4
5     def __init__(self, num_features):
6         super().__init__()
7         self.linear = torch.nn.Linear(num_features, 3)
8
9     def forward(self, x):
10        logits = self.linear(x)
11        probas = torch.nn.functional.softmax(logits)
12        return probas
```

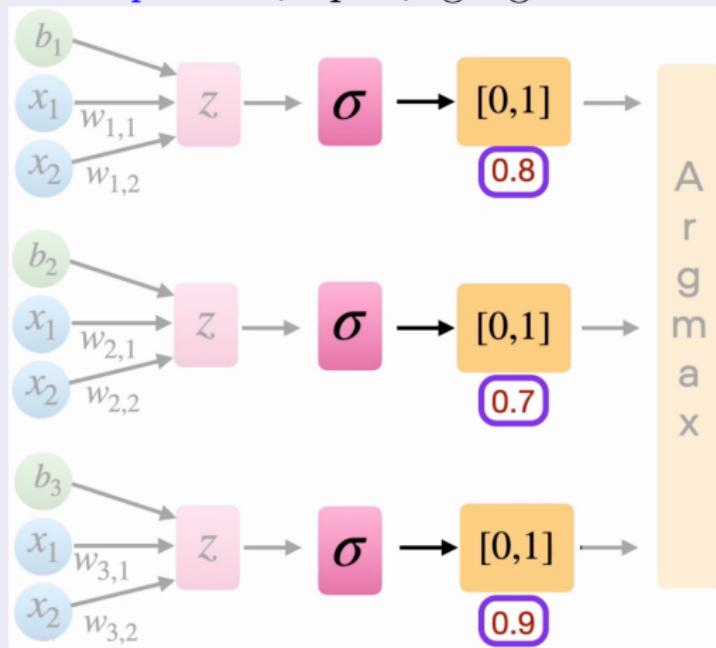
```
1 import torch
2
3+ class LogisticRegression(torch.nn.Module):
4
5     def __init__(self, num_features):
6         super().__init__()
7+         self.linear = torch.nn.Linear(num_features, 1)
8
9     def forward(self, x):
10        logits = self.linear(x)
11+       probas = torch.sigmoid(logits)
12        return probas
```

Lưu ý:

- ① Output của lớp linear là 3
- ② Hàm kích hoạt thay bằng softmax.

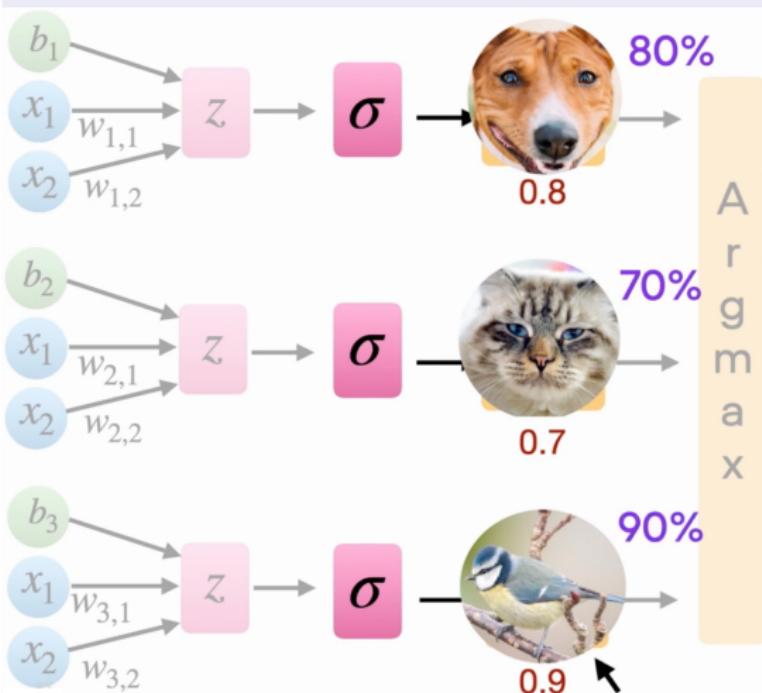
The Softmax Activation Function

Chúng ta có thể xem hàm kích hoạt *Softmax* là *dạng tổng quát của Sigmoid cho nhiều lớp*. Ví dụ áp dụng sigmoid cho 3 bộ trọng số:



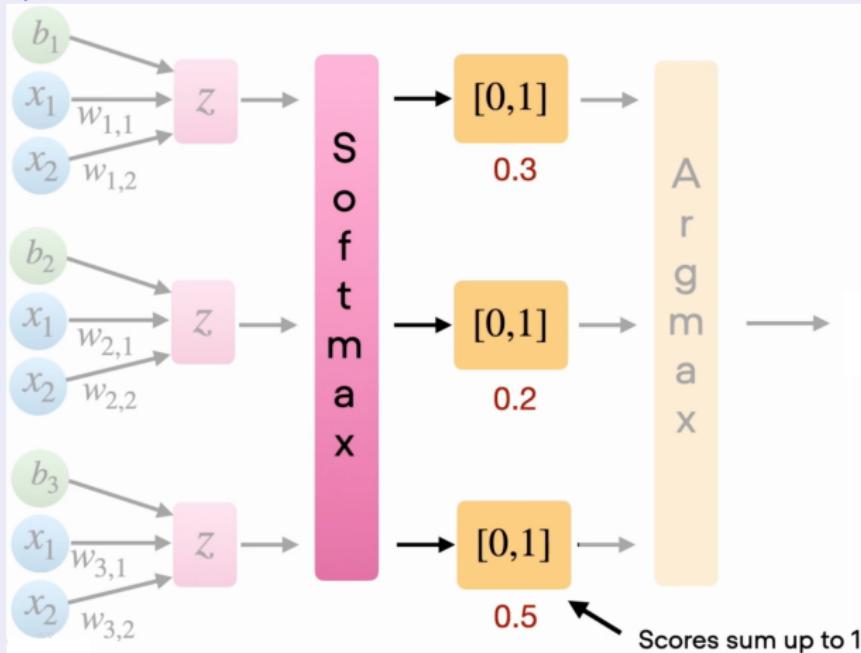
Tại sao không sử dụng sigmoid mà phải là softmax?

Rõ ràng, các xác suất 0.8, 0.7, 0.9 **không** có giá trị về mặt thống kê vì tổng những điểm xác suất của chúng không bằng 1.



Hình 2: Hãy thử hình dung, cho một training example, dự đoán xem training example đó thuộc vào lớp chó, mèo, chim với các xác xuất tương ứng 0.8, 0.7 và 0.9 có được không?

Do đó, chúng ta sẽ sử dụng Softmax để biến đổi các xác suất đầu vào của sigmoid thành một phân phối xác suất (*a probability distribution*)



Rescaling the probabilities so that they sum up to 1.

Công thức của Softmax activation

The mathematical formula for the softmax regression function

$$P(y = t | z_t^i) = \sigma_{softmax}(z_t^i) = \frac{e^{z_t^i}}{\sum_{j=1}^h e^{z_j^i}}. \quad (1)$$

Softmax score P is a class membership probability for a label t với $t \in [1, h]$, h là số lượng lớp. Công thức áp dụng cho một training example thứ i . $\sum_{j=1}^h e^{z_j^i}$ là hệ số chuẩn hoá (normalization term) để đảm bảo tổng xác suất bằng 1.

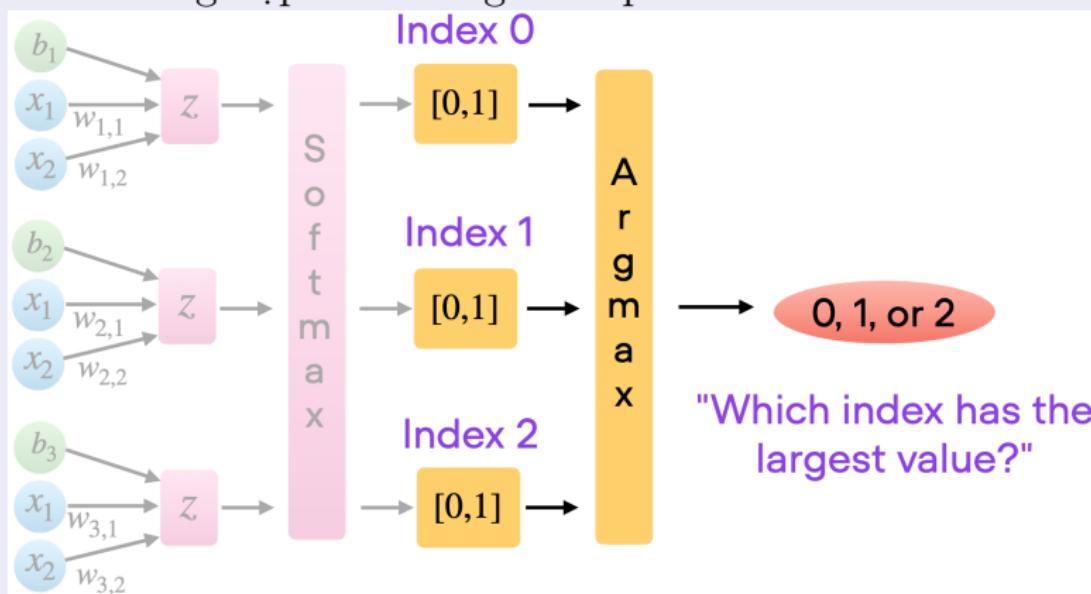
Ví dụ tính softmax cho 2 training examples, 3 classes

```
1 import torch
2 import torch.nn.functional as F
3 #net input
4 z = torch.tensor([[3.1, -2.3, 5.8], # 1st training example
5                  [1.1, 1.9, -8.9]]) # 2nd training example
6 torch.set_printoptions(sci_mode=False, precision=2) # Thiết lập hiển thị số
7          thập phân khi in
8 sm = F.softmax(z, dim=1) # dim =1 Áp dụng softmax theo hàng cho từng training
9          example
10 print(sm)
# tensor([[ 0.06,    0.0,    0.94],
#         [ 0.31,    0.69,    0.0]])
```

Argmax

From Softmax Scores to Class Labels

Từ các giá trị của softmax, ta sẽ chuyển nó về thành class labels.
Xem xét trường hợp 1 training example:



Argmax

Ví dụ tính softmax cho 2 training examples, 3 classes

```
sm = F.softmax(z, dim=1)
```

```
sm
```

```
tensor([[ 0.06, 0.00, 0.94], Example 1
        [ 0.31, 0.69, 0.00]])
```

```
tensor([[ 0.06, 0.00, 0.94], Index 2
        [ 0.31, 0.69, 0.00]])
```

Index 1

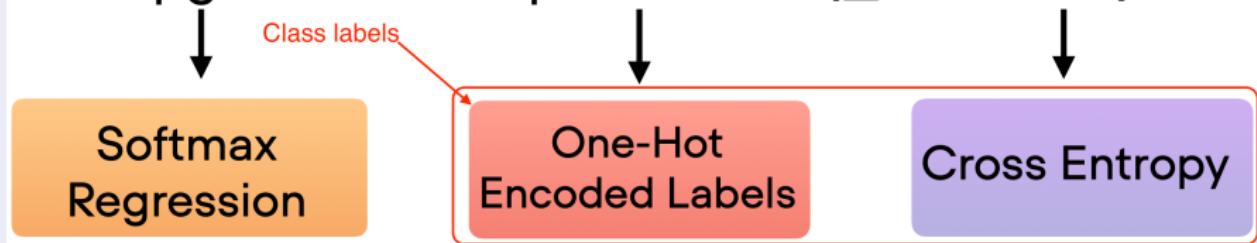
Sử dụng argmax function để chuyển từ softmax sang class label:

```
1 sm.argmax(dim=1) # tìm chỉ số có giá trị lớn nhất
2 # tensor([2, 1])
```

Cross Entropy Loss Function for Multiple Classes

Sau khi tìm được class labels từ hàm *argmax*, ta đi đến hàm Loss của bài toán softmax regression.

Upgrade to Multiple Classes (≥ 2 classes)



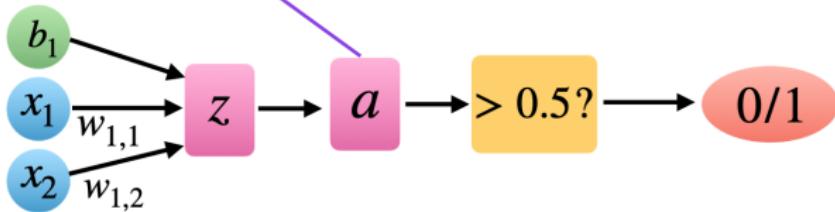
Cũng tương tự như Logistic regression, ta mở rộng Cross Entropy Loss từ 2 lớp cho trường hợp nhiều lớp với sự giúp đỡ của One-Hot encoded.

Binary cross entropy loss for **multiple** training examples

$$L = - \frac{1}{n} \sum_{i=1}^n - \left(y^{[i]} \log(a^{[i]}) + (1 - y^{[i]}) \log(1 - a^{[i]}) \right)$$

Class label, 0 or 1

Activation value (logistic sigmoid)



Binary Multi-category cross entropy loss for multiple training examples and multiple classes

$$L = -\frac{1}{n} \sum_{i=1}^n \sum_{k=1}^K -y_k^{[i]} \log(a_k^{[i]})$$

với K là số lượng class, $y_k^{[i]}$ là một a *one-hot encoding* cho traning example thứ i . Tại sao phải cần *one-hot encoding*?

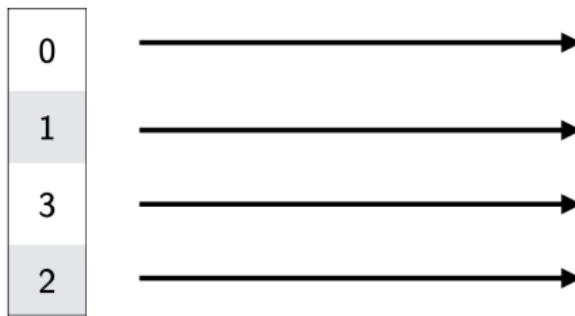
Trong trường hợp Binary, y chỉ nhận giá trị là 0 hoặc 1, trường hợp nhiều lớp thì $y \in \{0, 1, 2, \dots, K\}$. Vậy nên ta cần *one-hot encoding* để đưa các class label về 0, 1

Ví dụ thực hiện one-hot encoding

Xét trường hợp có 4 training example và 4 lớp như hình

Class labels

One-hot encoded class labels



Class 0	Class 1	Class 2	Class 3
1	0	0	0
0	1	0	0
0	0	0	1
0	0	1	0

Chúng ta thực hiện **chuyển** các class label 0, 1, 2, 3 của từng training example sang các one-hot value có giá trị là 0 và 1. Tại vị trí **one-hot index** tương ứng với class label của training example thứ i , ta cho bằng 1, tại các vị trí khác cho bằng 0.

Với $y_k^{[i]}$ là một *one-hot encoding* cho traning example thứ i , ta chỉ có duy nhất một one-hot value giá trị 1, các lớp khác có giá trị 0.

$$L = \frac{1}{n} \sum_{i=1}^n \sum_{k=1}^K -y_k^{[i]} \log(a_k^{[i]})$$

Assumes a one-hot encoding
(1's and 0's)

Dễ dàng nhận thấy rằng, *Multi-category Cross Entropy Loss-CE* là một dạng tổng quát của *Binary Cross Entropy Loss-BCE*.

Chứng minh

Với trường hợp có 2 lớp $K = 2$, $y_k^{[i]}$ được biểu diễn bằng một one-hot vector có dạng $[y_1^i, y_2^i] \in \{0, 1\}$. Chúng ta có thể viết lại one-hot vector dưới dạng $[y^i, 1 - y^i]$, $y^i \in \{0, 1\}$. Thay vào trong công thức của *CE*:

$$L = \frac{1}{n} \sum_{i=1}^n -y^i \log(a_1^i) - (1 - y^i) \log(a_2^i),$$

trong đó, $a_1^i = a$, $a_2^i = 1 - a$. Vậy ta thu được

$$L = \frac{1}{n} \sum_{i=1}^n -y^i \log(a) - (1 - y^i) \log(1 - a) \quad \blacksquare (đpcm)$$

Softmax regression in PyTorch

Ví dụ bài toán Softmax regression cho bài toán phân loại có 3 lớp, 4 training examples sử dụng PyTorch.

One-hot encoding in PyTorch

Trong PyTorch, để tính one-hot encoding cho các lớp ta sử dụng `torch.nn.functional.one_hot()`

```
1 import torch
2 import torch.nn.functional as F
3
4 y = torch.tensor([0,2,2,1])
5 y_onehot = F.one_hot(y)
6 print(y_onehot)
7 #tensor([[1, 0, 0],
8 #        [0, 0, 1],
9 #        [0, 0, 1],
10 #       [0, 1, 0]])
```

Softmax regression in PyTorch

Tính *softmax activation* cho net _inputs z trong PyTorch

```
1 net_inputs = torch.tensor([[1.5, 0.1, -0.4],  
2                             [0.5, 0.7, 2.1],  
3                             [-2.1, 1.1, 0.8],  
4                             [1.1, 2.5, -1.2]])  
5 softmax_activation = F.softmax(net_inputs, dim=1)  
6 print(softmax_activation)  
7 torch.sum(softmax_activation, dim=1)  
8 # tensor([0.7162, 0.1766, 0.1071],  
9 #           [0.1394, 0.1702, 0.6904],  
10 #          [0.0229, 0.5613, 0.4158],  
11 #          [0.1940, 0.7866, 0.0194]])  
12 # tensor([1.0000, 1.0000, 1.0000, 1.0000])
```

Softmax regression in PyTorch

Tính bằng tay Loss dựa trên công thức

$L = \frac{1}{n} \sum_{i=1}^n \sum_{k=1}^K -y_k^i \log(a_k^i)$. Biết rằng ta có:

- softmax activation a

```
1 # tensor([[0.7162, 0.1766, 0.1071],  
2 #           [0.1394, 0.1702, 0.6904],  
3 #           [0.0229, 0.5613, 0.4158],  
4 #           [0.1940, 0.7866, 0.0194]])
```

- one-hot encoding true labels y

```
1 #tensor([[1, 0, 0],  
2 #         [0, 0, 1],  
3 #         [0, 0, 1],  
4 #         [0, 1, 0]])
```

1 # Kết quả tính Loss cho 4 training example:
2 L = (-1log(0.7162) -1log(0.6904) -1log(0.4158) -1log(0.7866))/4 = 0.4555

Softmax regression in PyTorch

Tính Loss dựa trên code PyTorch

```
1 import torch
2 import torch.nn.functional as F
3
4 def manual_cross_entropy(net_inputs,y):
5     softmax_activation = F.softmax(net_inputs,dim=1)
6     y_onehot = F.one_hot(y)
7     L= torch.mean(torch.sum(-y_onehot*torch.log(softmax_activation),dim=1))
8     return L
9
10 y = torch.tensor([0,2,2,1])
11 net_inputs = torch.tensor([[1.5, 0.1, -0.4],
12                           [0.5, 0.7, 2.1],
13                           [-2.1, 1.1, 0.8],
14                           [1.1, 2.5, -1.2]])
15 manual_cross_entropy(net_inputs,y)
16 # tensor(0.4555)
```

Kết quả thu được tương tự như tính tay ở trên.

Softmax regression in PyTorch

Trong thực tế, chúng ta không cần thực hiện các one-hot encoding, softmax activation khi tính Loss. Trong PyTorch hỗ trợ hàm `torch.nn.functional.cross_entropy()` để tính Loss

```
1 import torch
2 import torch.nn.functional as F
3
4 y = torch.tensor([0,2,2,1])
5 net_inputs = torch.tensor([[1.5, 0.1, -0.4],
6                           [0.5, 0.7, 2.1],
7                           [-2.1, 1.1, 0.8],
8                           [1.1, 2.5, -1.2]])
9 F.cross_entropy(net_inputs,y)
# tensor(0.4555)
```

Lưu ý

Một vài hàm cross-entropy loss trong PyTorch:

- *torch.nn.functional.binary_cross_entropy* takes logistic sigmoid values as inputs
- *torch.nn.functional.binary_cross_entropy_with_logits* takes logits as inputs
- *torch.nn.functional.cross_entropy* takes logits as inputs (performs log_softmax internally)
- *torch.nn.functional.nll_loss* is like cross_entropy but takes log-probabilities (log-softmax) values as inputs

Tài liệu tham khảo

-  Sebastian Raschka, Yuxi (Hayden) Liu, Vahid Mirjalili
Machine Learning with PyTorch and Scikit-Learn: Develop machine learning and deep learning models with Python (2022). Published by Packt Publishing Ltd, ISBN 978-1-80181-931-2.
-  Sebastian Raschka
MACHINE LEARNING Q AND AI: 30 Essential Questions and Answers on Machine Learning and AI (2024). ISBN-13: 978-1-7185-0377-9 (ebook).
-  LightningAI
LightningAI: PyTorch Lightning (2024) .