

Multiclass classification with neural networks

Softmax regression

Filippo Biscarini
Senior Scientist
CNR, Milan (Italy)

Nelson Nazzicari
Senior Scientist
CREA, Lodi (Italy)



Multiclass classification



(a)



(b)



(c)



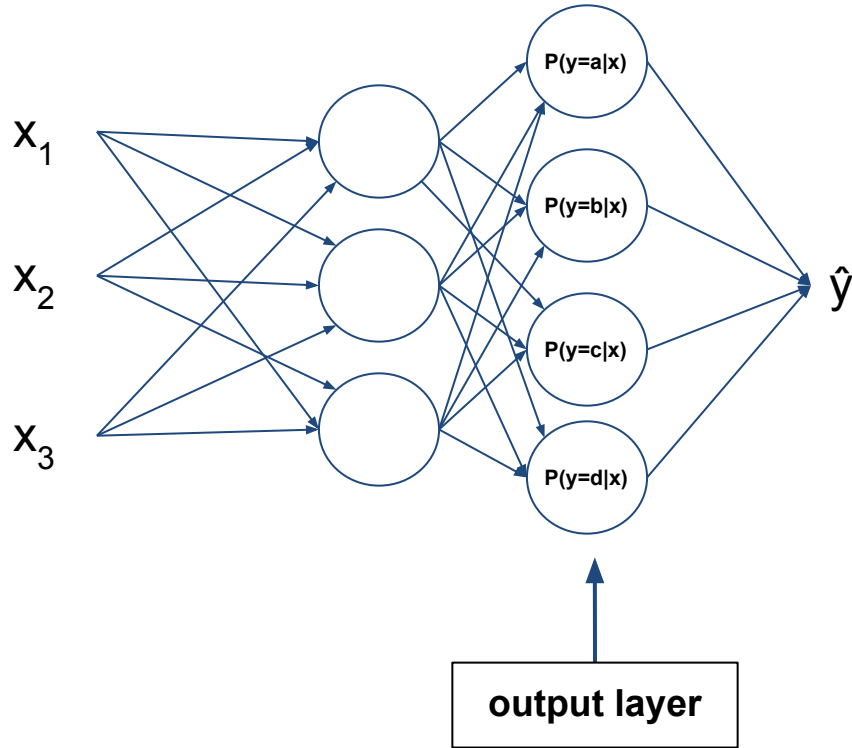
(d)

From: Wang et al., 2015 (Entropy)

- **more than two classes** to recognize
- in this case, 4 classes: bananas, green apples, red apples, oranges
- need to extend logistic regression to estimating the **probabilities** of samples **to belong** to each of the **four classes**



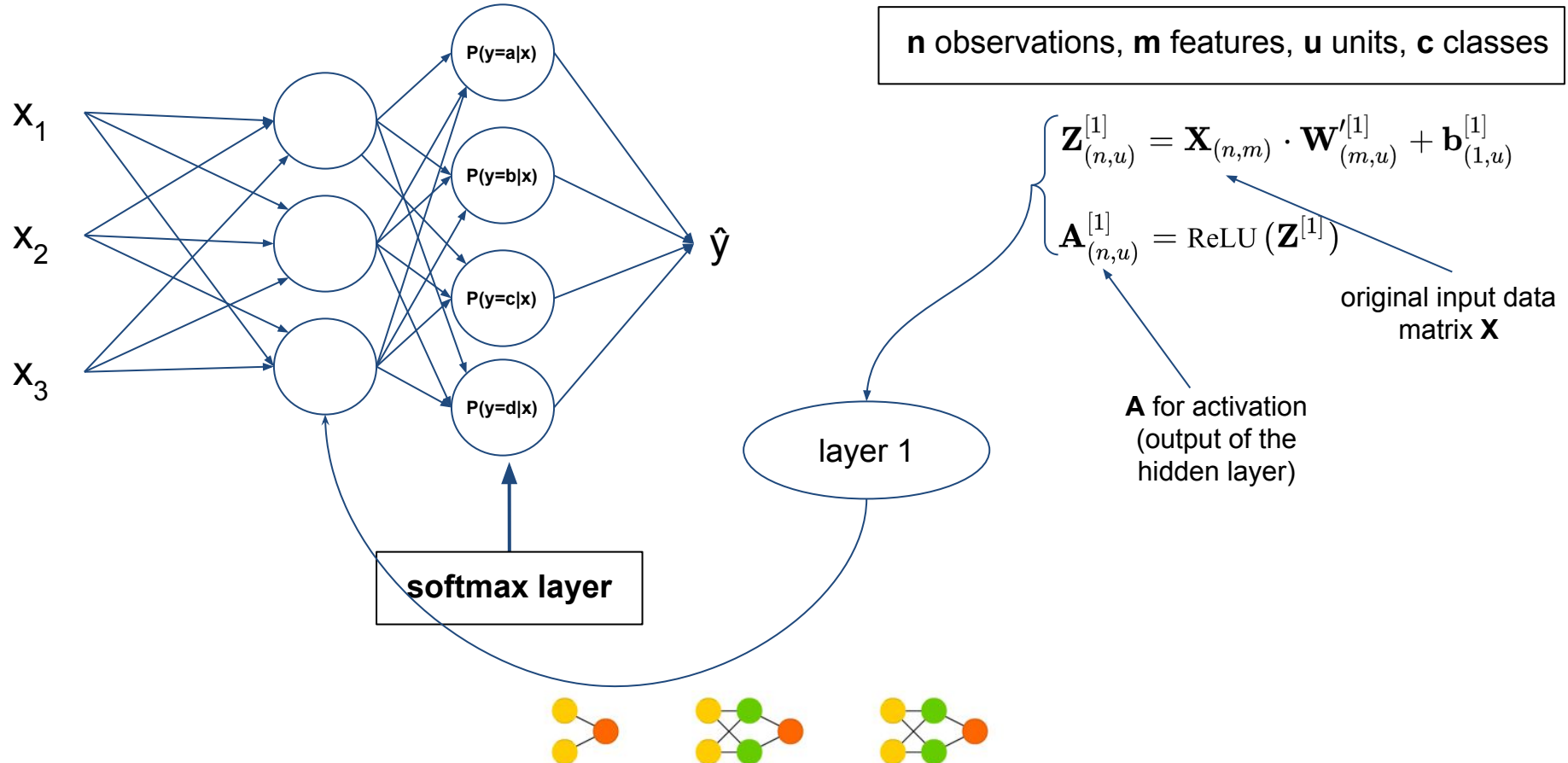
Multiclass classification



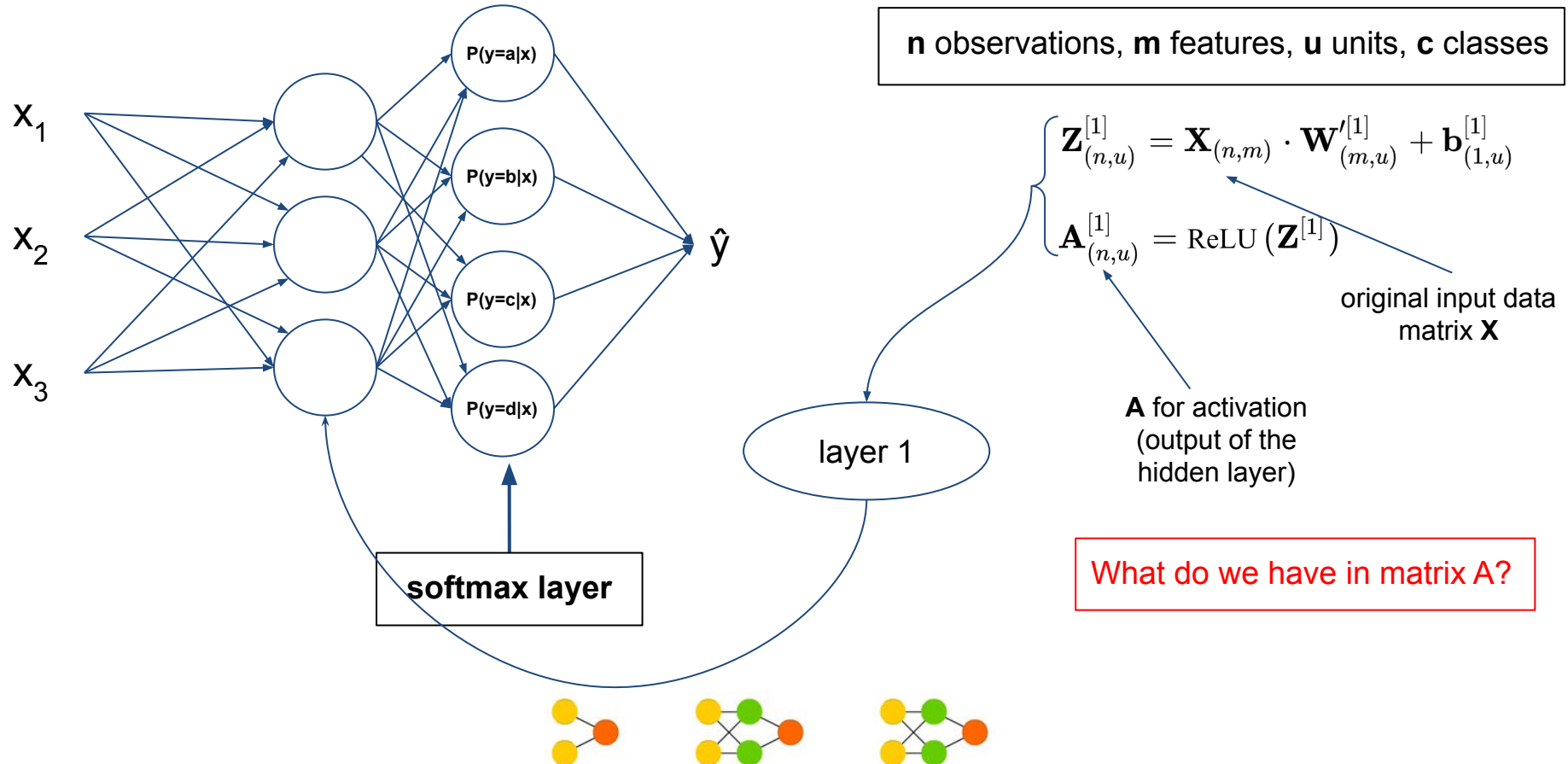
- the **output layer** has now **4 nodes** (instead of just one as in binary classification)
- the **activation** function is now the **softmax function**
- this is why the output layer is also called the **softmax layer**, and multiclass classification is called **softmax regression**



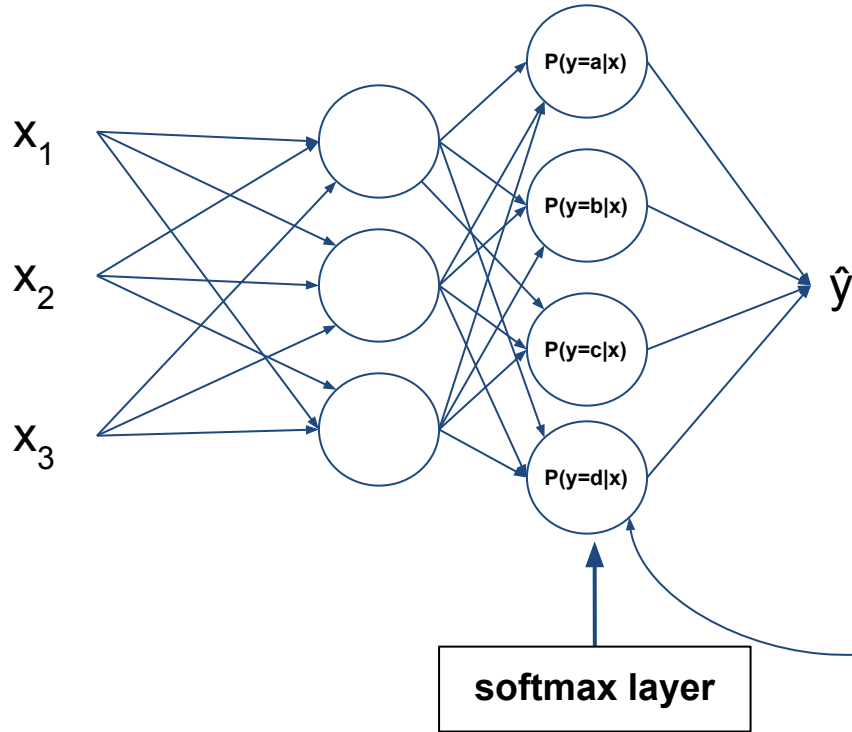
Multiclass classification



Multiclass classification



Multiclass classification



n observations, **m** features, **u** units, **c** classes

$$\begin{cases} \mathbf{Z}_{(n,u)}^{[1]} = \mathbf{X}_{(n,m)} \cdot \mathbf{W}_{(m,u)}'^{[1]} + \mathbf{b}_{(1,u)}^{[1]} \\ \mathbf{A}_{(n,u)}^{[1]} = \text{ReLU}(\mathbf{Z}^{[1]}) \end{cases}$$

$$\mathbf{Z}_{(n,c)}^{[2]} = \mathbf{A}_{(n,u)}^{[1]} \cdot \mathbf{W}_{(u,c)}'^{[2]} + \mathbf{b}_{(1,c)}^{[2]}$$

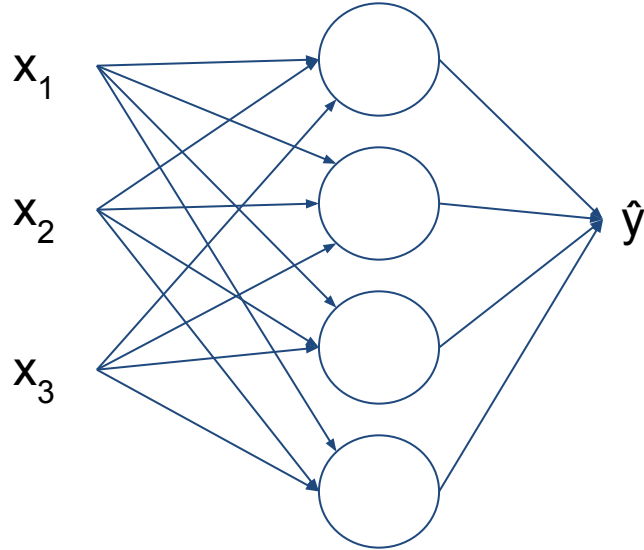
$$\hat{\mathbf{y}}_{(n,c)} = \text{softmax}(\mathbf{Z}^{[2]})$$

softmax activation

c, classes: $\hat{\mathbf{y}}$ is now a matrix (4 probabilities per sample)



Multiclass classification



- as we did for binary classification, we can also represent softmax regression with no hidden layers
- this would essentially reduce to multinomial logistic regression, with no neural networks involved
- we'll see this implementation in the ipynb file `day3_block01`, part 1



Loss function for softmax regression

$$L(\hat{y}, y) = -(y \cdot \log(\hat{y}) + (1 - y) \cdot \log(1 - \hat{y}))$$

loss function for logistic
regression

$$L(\hat{y}, y) = - \sum_{j=1}^c y_j \cdot \log(\hat{y}_j)$$

loss function for softmax
regression

- generalization of the loss function for logistic regression over c classes
- you then sum up the loss for each sample (and divide by the number of samples) → cost function for the entire dataset



Loss function for softmax regression

$$L(\hat{y}, y) = -(y \cdot \log(\hat{y}) + (1 - y) \cdot \log(1 - \hat{y}))$$

loss function for logistic
regression

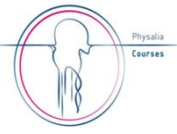
$$L(\hat{\mathbf{y}}, \mathbf{y}) = - \sum_{j=1}^c y_j \cdot \log(\hat{y}_j)$$

loss function for softmax
regression

these are actually vectors of length c (number of classes)!



Recap



- we can represent logistic and softmax regression as neural networks (NN) models
- when we add layers and units we **repeat several times the regression model** with a twist given by the ReLU activation
- these multiple regression models all use the same input data, but **are all different** because they learn different weights (parameters)
- as we progress along the layers, we get farther and farther from the original input data → we learn new (more abstract) representations of the data
- when we “go deep” we move away from the linear decision boundaries of logistic (and softmax regression) and **learn complex non-linear functions of the data**



Multiclass classification

- multiclass logistic regression
- neural network for softmax regression

→ day3_code01 keras multiclass classification.ipynb

