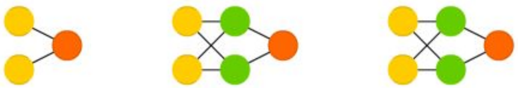
Multiclass classification with neural networks

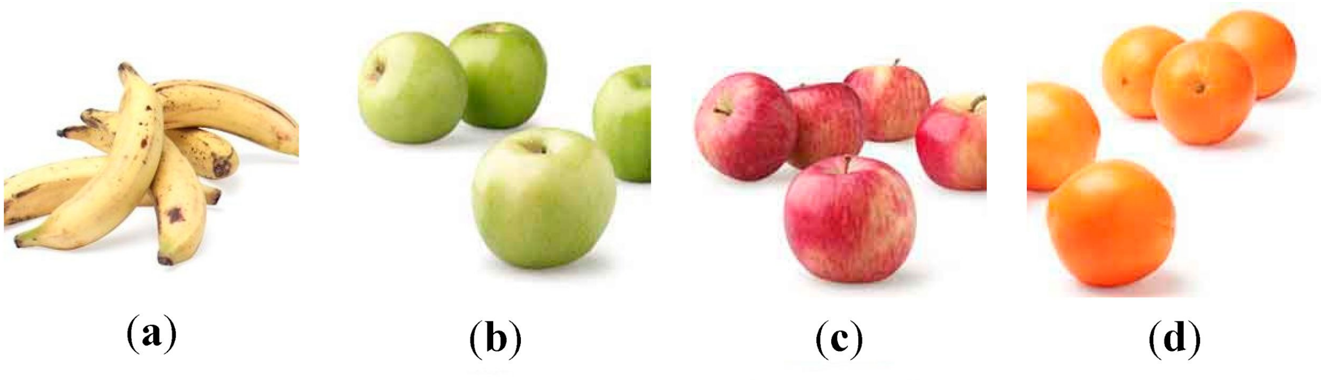
Softmax regression

Filippo Biscarini
Senior Scientist
CNR, Milan (Italy)

Nelson Nazzicari
Research fellow
CREA, Lodi (Italy)



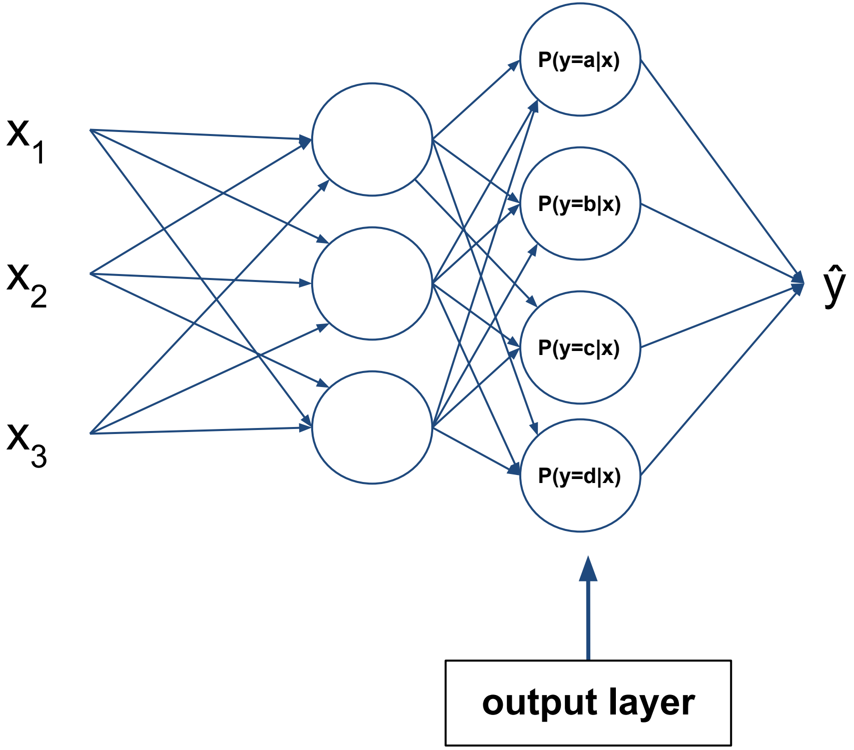
Multiclass classification



From: Wang et al., 2015 (Entropy)

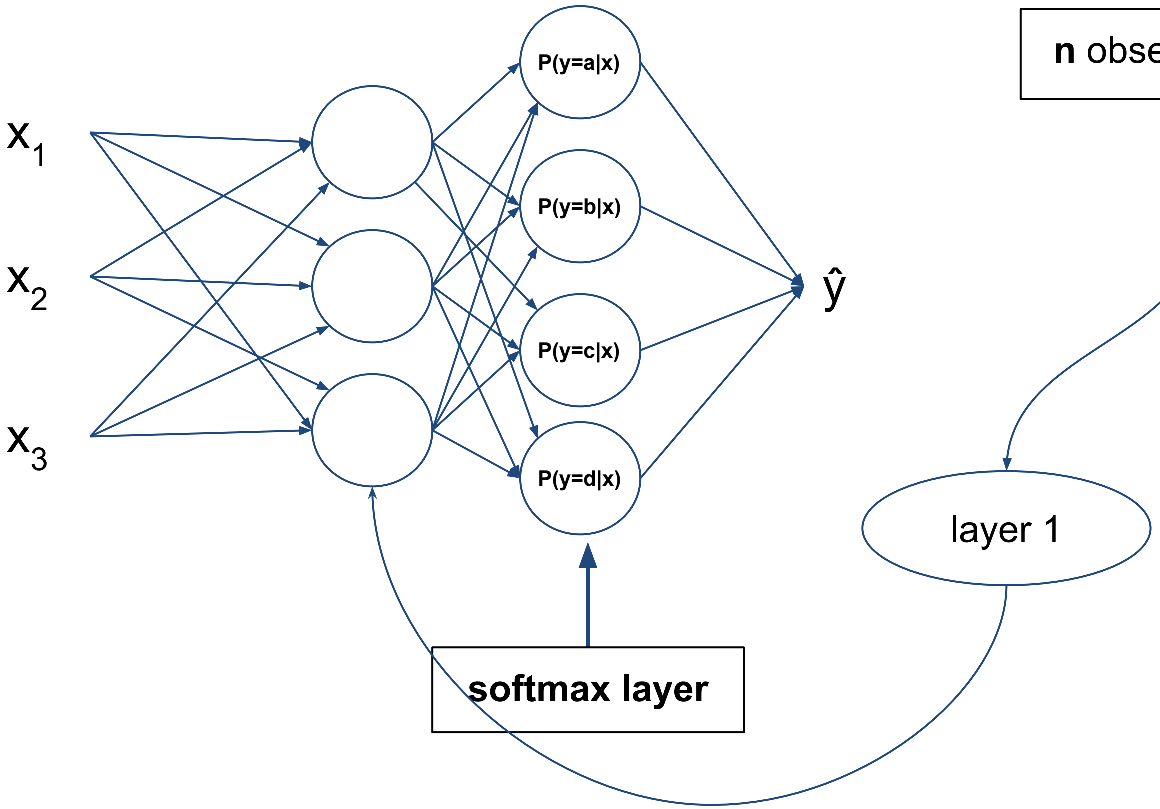
- **more than two classes** to recognize
- in this case, 4 classes: bananas, green apples, red apples, oranges
- need to extend logistic regression to estimating the **probabilities** of samples **to belong** to each of the **four classes**

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

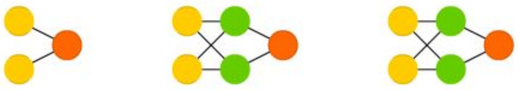


n observations, **m** features, **u** units, **c** classes

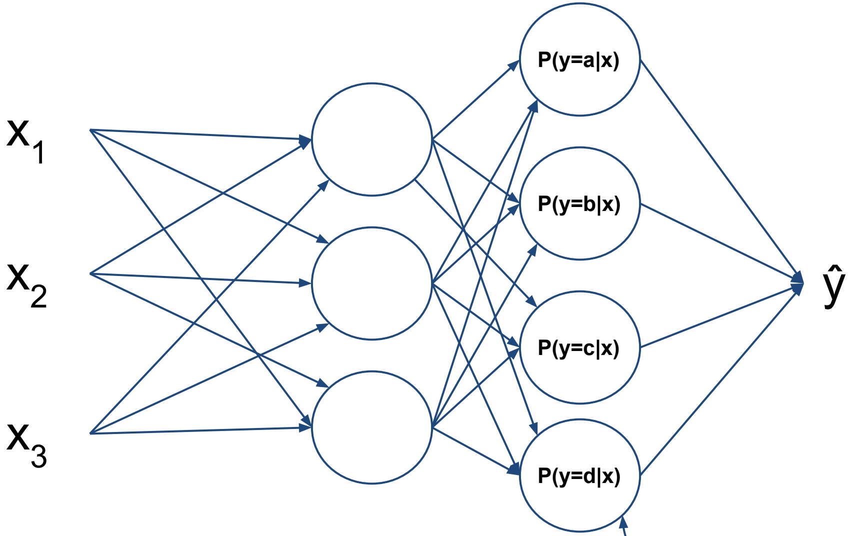
$$\begin{cases} \mathbf{Z}^{[1]}_{(n,u)} = \mathbf{X}_{(n,m)} \cdot \mathbf{W}'^{[1]}_{(m,u)} + \mathbf{b}^{[1]}_{(1,u)} \\ \mathbf{A}^{[1]}_{(n,u)} = \mathrm{ReLU}\left(\mathbf{Z}^{[1]}\right) \end{cases}$$

original input data matrix **X**

A for activation (output of the hidden layer)



Multiclass classification



layer 2

softmax layer

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

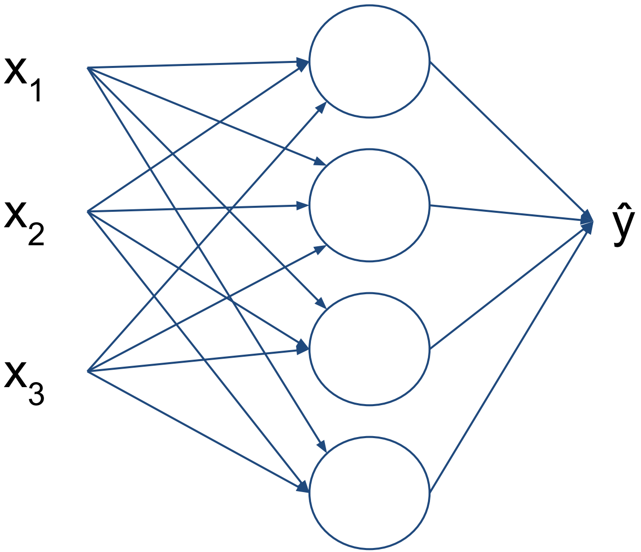
$$\begin{cases} \mathbf{Z}^{[1]}_{(n,u)} = \mathbf{X}_{(n,m)} \cdot \mathbf{W}'^{[1]}_{(m,u)} + \mathbf{b}^{[1]}_{(1,u)} \\ \mathbf{A}^{[1]}_{(n,u)} = \mathrm{ReLU}\left(\mathbf{Z}^{[1]}\right) \end{cases}$$

$$\begin{cases} \mathbf{Z}^{[2]}_{(n,c)} = \mathbf{A}^{[1]}_{(n,u)} \cdot \mathbf{W}'^{[2]}_{(u,c)} + \mathbf{b}^{[2]}_{(1,c)} \\ \hat{\mathbf{y}}_{(n,c)} = \mathrm{softmax}\left(\mathbf{Z}^{[2]}\right) \end{cases}$$

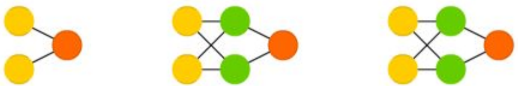
softmax activation

c, classes: ŷ 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 4c, part 1



Loss function for softmax regression

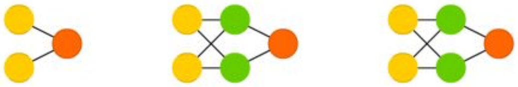
$$L(\hat{y}, y) = -\left(y \cdot \log\left(\hat{y}\right) + (1 - y) \cdot \log\left(1 - \hat{y}\right)\right)$$

loss function for logistic regression

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

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

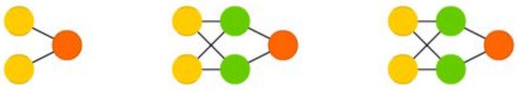
$$L(\hat{y}, y) = -\left(y \cdot \log\left(\hat{y}\right) + \left(1 - y\right) \cdot \log\left(1 - \hat{y}\right)\right)$$

loss function for logistic regression

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

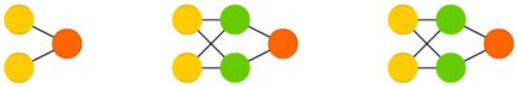
loss function for softmax regression

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



Recap

- we can represent logistic and softmax regression as neural networks 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

- demonstration 04c
- exercise 04c.1 (multiclass logistic regression)

→ code_04c_keras_multiclass_classification.ipynb

Neural networks models: recap

- exercise 04d.1 (write your own code)

  → code_04d_neural_networks_exercise.ipynb