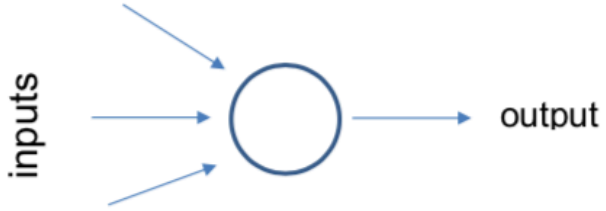
Neural Networks - II

The perceptron

- This a model of a neuron (the building block on the NN)
- Mathematically, it is a transformation on the inputs (predictors) x_i which is producing an output x_o

$$x_o = f_o\left(\sum_{inputs:i} w_i x_i\right)$$



The terminology is somewhat different in NN:

- f_o is called an activation function (can be identity, logistic, indicator…)
- w_i are called weights instead of coefficients (if $x_1 \equiv 1$ then the corresponding weight w_1 is called a bias – it's just an intercept)
- The NN learns the weights from the data

Regression Multulayer Perceptrons (MLPs)

- MLPs can be used for regression tasks. If you want to predict a single value (e.g., the price of a house, given many of its features), then you just need a single output neuron: its output is the predicted value.

- For multiple regression (i.e., to predict multiple values at once), you need 1 output neuron per output dimension. For example, to locate the center of an object in an image, you need to predict 2D coordinates, so you need two output neurons.

- In general, when building an MLP for regression, you do not want to use any activation function for the output neurons, so they are free to output any range of values.

- if you want to guarantee that the predictions will fall within a given range of values, then you can use the logistic function or the hyperbolic tangent, and then scale the labels to the appropriate range:
    - 0 to 1 for the logistic function
    - –1 to 1 for the hyperbolic tangent.

- The loss function is typically the MSE. If you have a lot of outliers mean absolute error (MAE) instead.

Hyperparameter	Typical value
# input neurons	One per input feature (e.g., 28 x 28 = 784 for MNIST)
# hidden layers	Depends on the problem, but typically 1 to 5
# neurons per hidden layer	Depends on the problem, but typically 10 to 100
# output neurons	1 per prediction dimension
Hidden activation	ReLU (or SELU, see Chapter 11)
Output activation	None, or ReLU/softplus (if positive outputs) or logistic/tanh (if bounded outputs)
Loss function	MSE or MAE/Huber (if outliers)

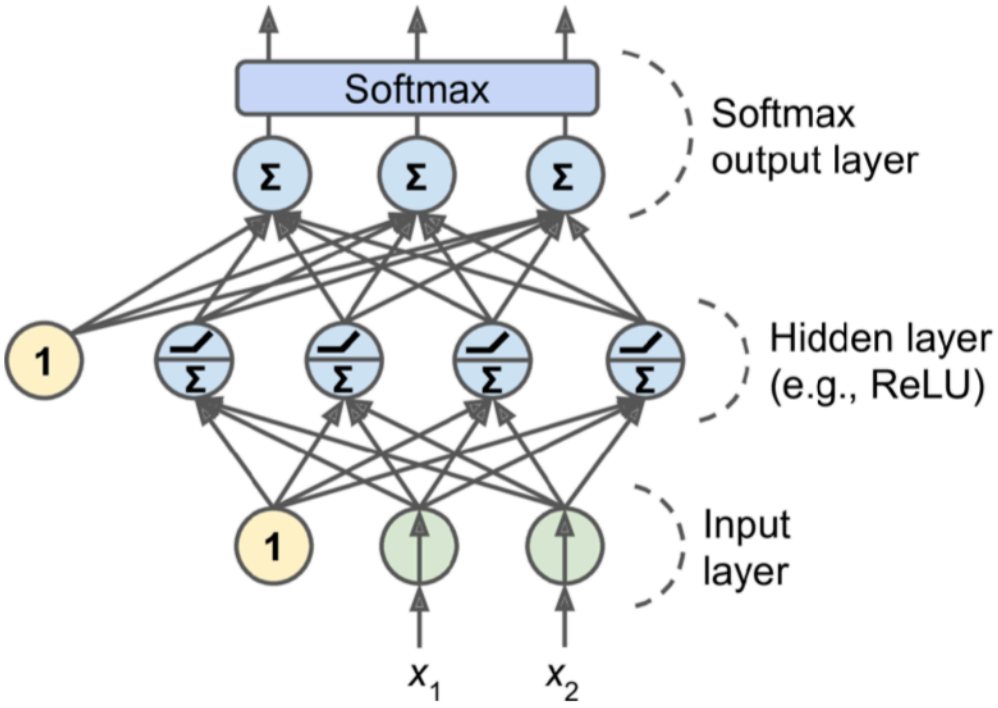
Classification MLPs

- For a binary classification problem, you just need a single output neuron using the logistic activation function: the output will be a number between 0 and 1, which you can interpret as the estimated probability of the positive class. The estimated probability of the negative class is equal to one minus that number.

- MLPs can also easily handle multilabel binary classification tasks. E.g., you could have an email classification system that predicts whether each incoming email is “ham” or spam, and simultaneously predicts whether it is an urgent or nonurgent email => 2 output neurons, both using the logistic activation function: the first would output the probability that the email is spam, and the second would output the probability that it is urgent.

  The output probabilities do not necessarily add up to 1. This lets the model output any combination of labels: you can have nonurgent ham, urgent ham, etc.

- Multiclass classification
  If each instance can belong only to a single class (# classes >=3, then you need to have one output neuron per class, and you should use the softmax activation function for the whole output layer. The softmax function will ensure that all the estimated probabilities are between 0 and 1 and that they add up to 1 (which is required if the classes are exclusive)



Typical values of the hyperparameters

Hyperparameter	Binary classification	Multilabel binary classification	Multiclass classification
Input and hidden layers	Same as regression	Same as regression	Same as regression
# output neurons	1	1 per label	1 per class
Output layer activation	Logistic	Logistic	Softmax
Loss function	Cross entropy	Cross entropy	Cross entropy

Examples in DSML13_NN2.ipynb