# Machine Learning Using Python

## Lesson 10: Neural Networks

---

Marcel Scharth

The University of Sydney Business School

# Neural Networks

**Neural networks** are a powerful class of nonlinear learning methods.

Neural networks learn complex combinations of the original inputs as derived features, and then model the response as a nonlinear function of these features.

# Single Layer Perceptron

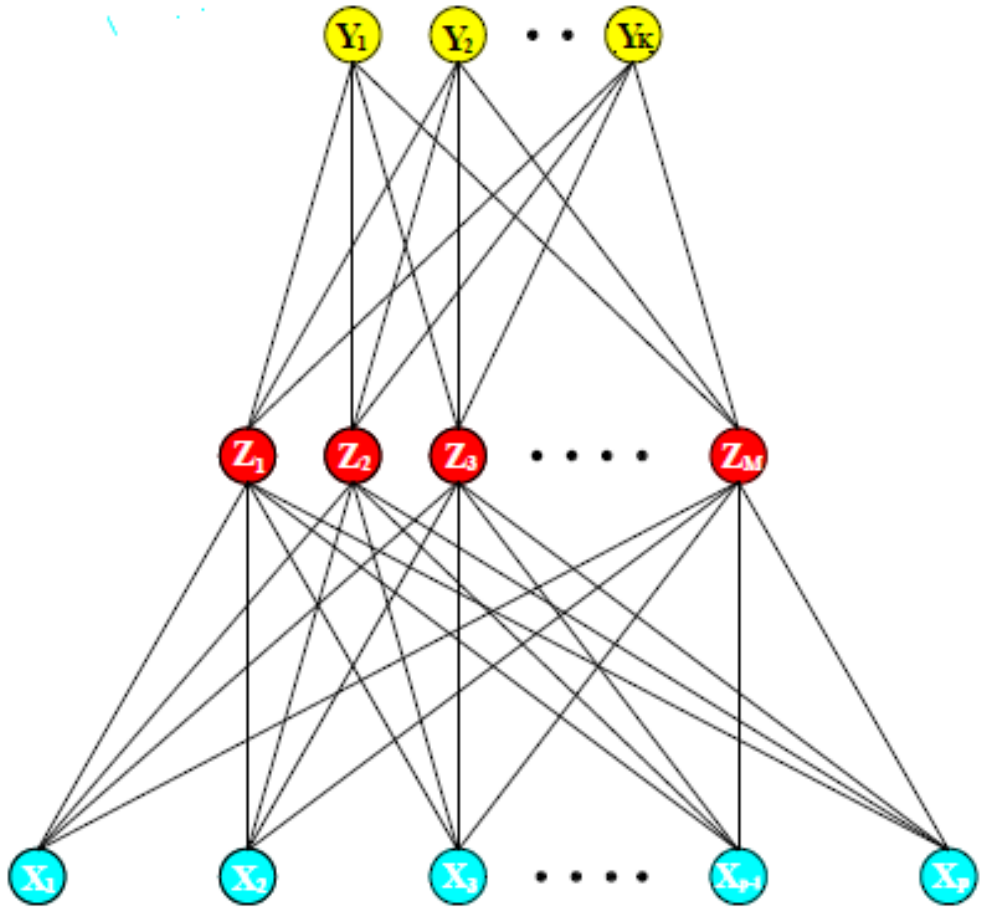



FIGURE 11.2. *Schematic of a single hidden layer, feed-forward neural network.*

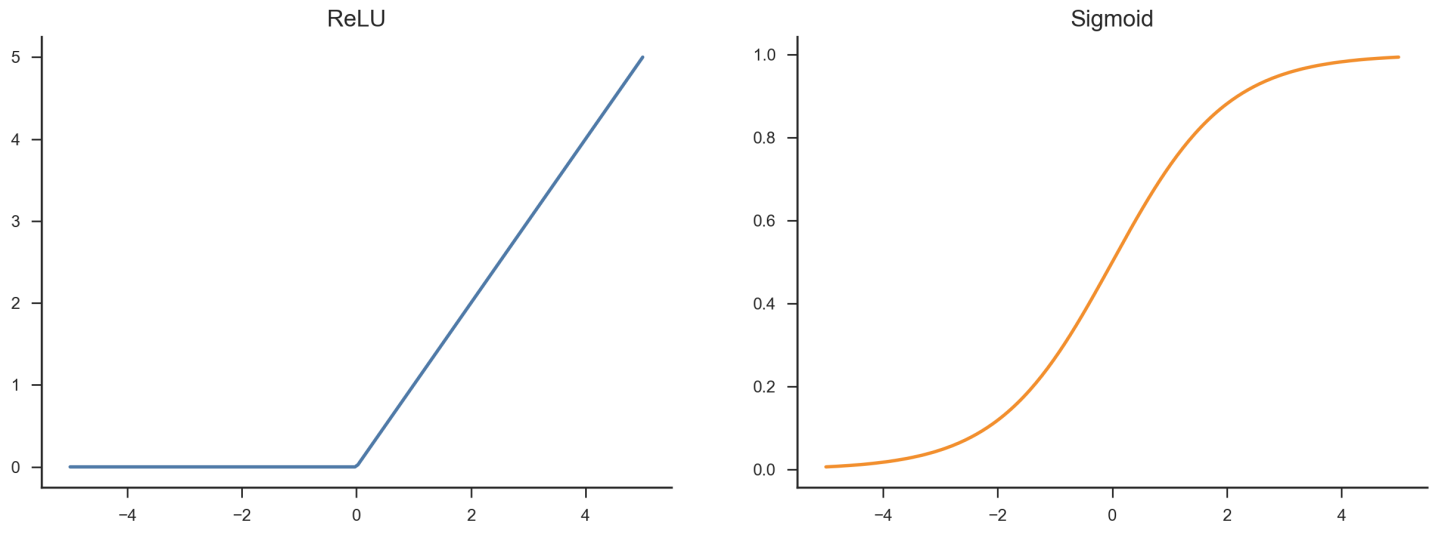
# Single layer perceptron

The **single layer perceptron** (or single hidden layer network) model is

$$
\begin{aligned}
Z_m &= \sigma\left(w_{0m} + w_{1m}x_1 + \ldots + w_{pm}x_p\right), \quad m = 1, \ldots, M, \\
T &= \beta_0 + \beta_1 Z_1 + \beta_2 Z_2 + \ldots + \beta_m Z_m, \\
f(\boldsymbol{x}) &= g(T),
\end{aligned}
$$

where $\sigma(\cdot)$ is an **activation function**, $Z_1, \ldots, Z_M$ are the **hidden units**, and $g(\cdot)$ is an output function.

# Activation Functions



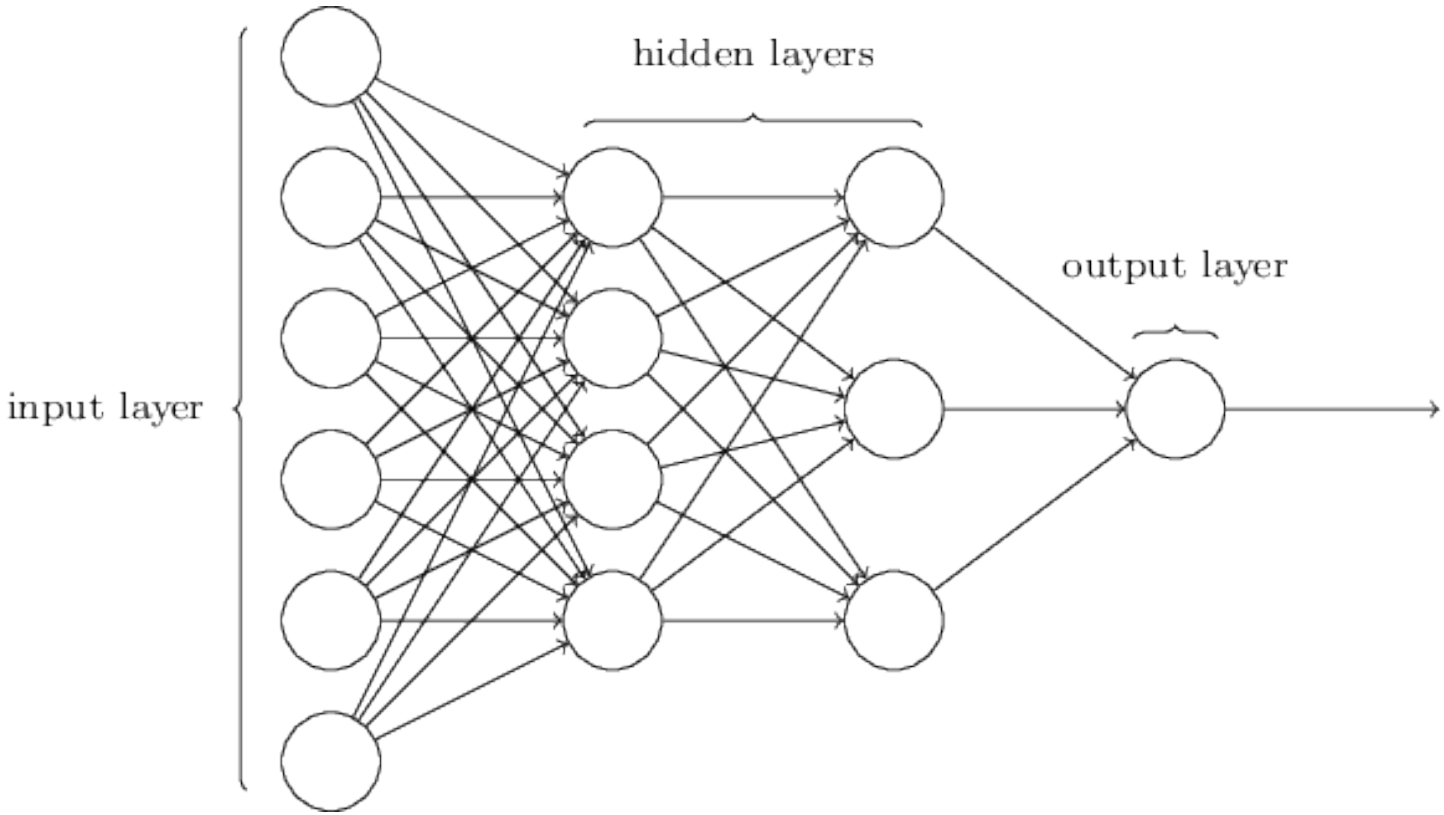

The rectifier linear unit (ReLU) activation is $\max(z, 0)$.

# Multilayer perceptron

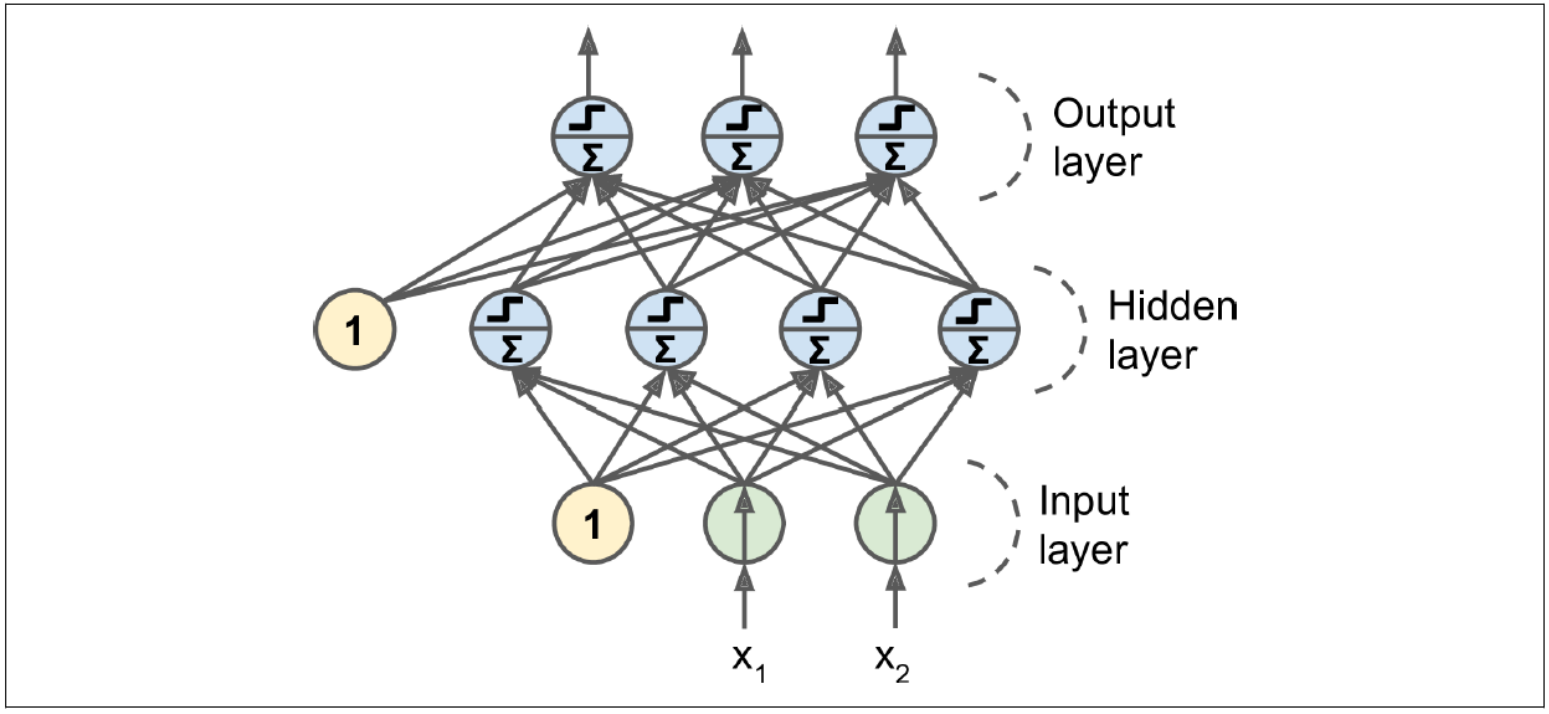
The **multilayer perceptron** (or **feedforward neural network**) is composed of an input layer, one or more layers of hidden units, and a final output layer.

A MLP with two or more hidden layers is called a **deep neural network**

# Multilayer Perceptron



# Multilayer Perceptron (multiple classes)



# Neural network hyperparameters

- **Number of hidden layers**: even though a single layer perceptron can model very complex functions with enough neurons, deep networks have much higher parameter efficiency. For many problems with start with one or two hidden layers, and it will be sufficient.

- **Number of neurons for each layer**: more neurons lead to a more flexible neural networks. One approach is to specify a model with many neurons and techniques to prevent overfitting.

- **Activation functions**: typically ReLU activations or variants for the hidden layers.

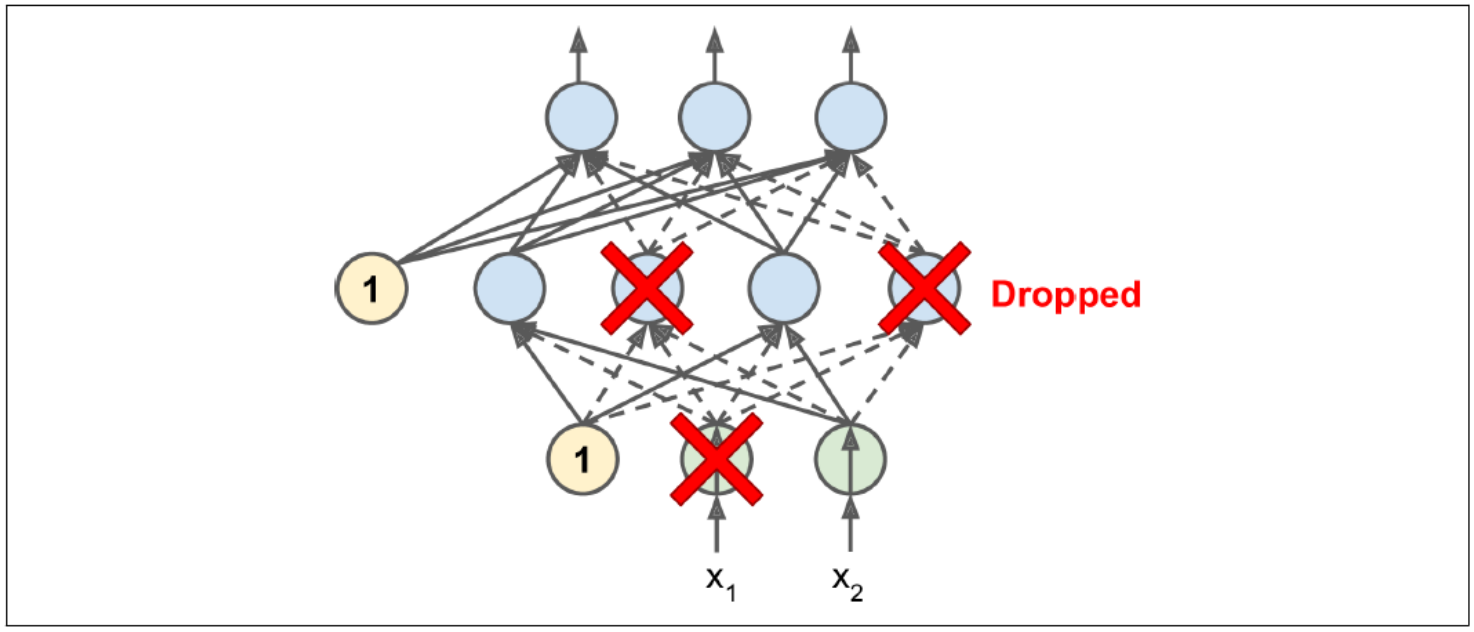
# Avoiding overfitting

- **Early stopping** interrupts training the performance on a validation set stops to improve and stars dropping. It works well in practice, though it is usually better to combine it with other regularisation techniques.

- $\ell_1$ and $\ell_2$ regularisation of the connection weights.

- **Dropout**. At every training step, every unit has a certain probability $p$ (typically 50%) of being entirely ignored during that training step (though it may be active in the next). Very popular in deep learning, and works well.

# Dropout



# Training neural networks

- Training neural networks can be very challenging and extremely slow. The several issues that arise are outside the scope of our lessons. Progress in this area has been instrumental to growth in the use of deep learning.

- As a practical detail, it is necessary to scale the predictors.

# Training neural networks: overview of needed configuration

- Initialisation.
- Activation function.
- Normalisation.
- Regularisation.
- Optimisation.
- Learning rate schedule.

# Training neural networks: guidelines

A configuration that may work well in most cases is:

- Initialisation: He initalisation.
- Activation function: ELU.
- Normalisation: batch normalisation.
- Regularisation: dropout.
- Optimisation: Nesterov Accelerated Gradient.
- Learning rate schedule: none.

These practical guidelines are suggested Geron (2017), referenced in the workshop website.