1. Biological vs. Artificial Neurons

- **Biological Neurons:**
  - Receive inputs (signals) from other neurons.
  - Process these inputs and produce an output (signal).

Understanding perceptrons

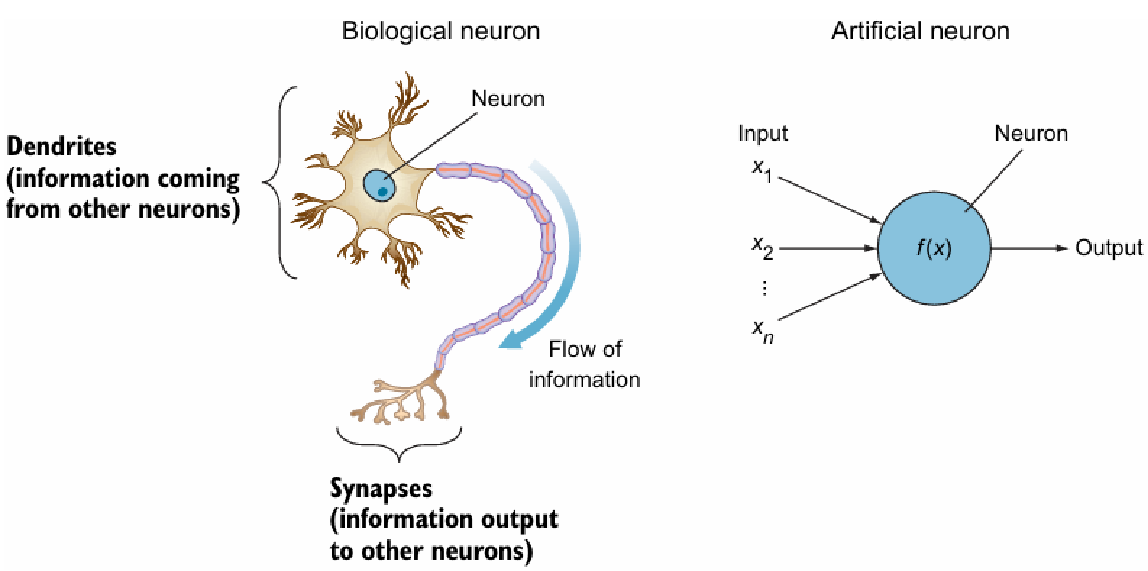


Figure 2.3 Artificial neurons were inspired by biological neurons. Different neurons are connected to each other by synapses that carry information.

- **Artificial Neurons:**
  - Mimic biological neurons.
  - Take inputs, apply calculations (weighted sum + activation function), and produce an output.

2. Structure of an Artificial Neuron

- **Inputs:** Features or data points (e.g., $x1,x2,x3$).
- **Weights:** Each input is multiplied by a weight (w1 ,w2 ,w3) that represents its importance.
- **Weighted Sum:** $z=w1x1+w2x2+w3x3+bias$
- **Activation Function:** Applies a non-linear transformation to the weighted sum to produce the output. $Output=f(z)$

3. Single Perceptron

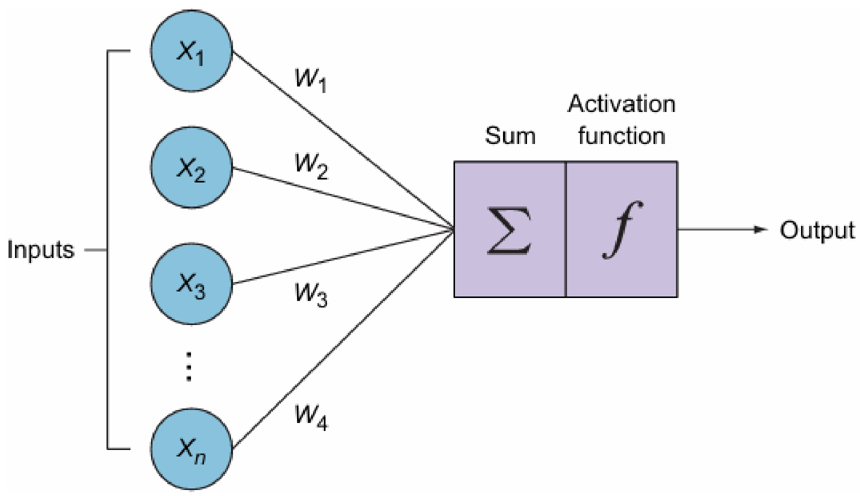
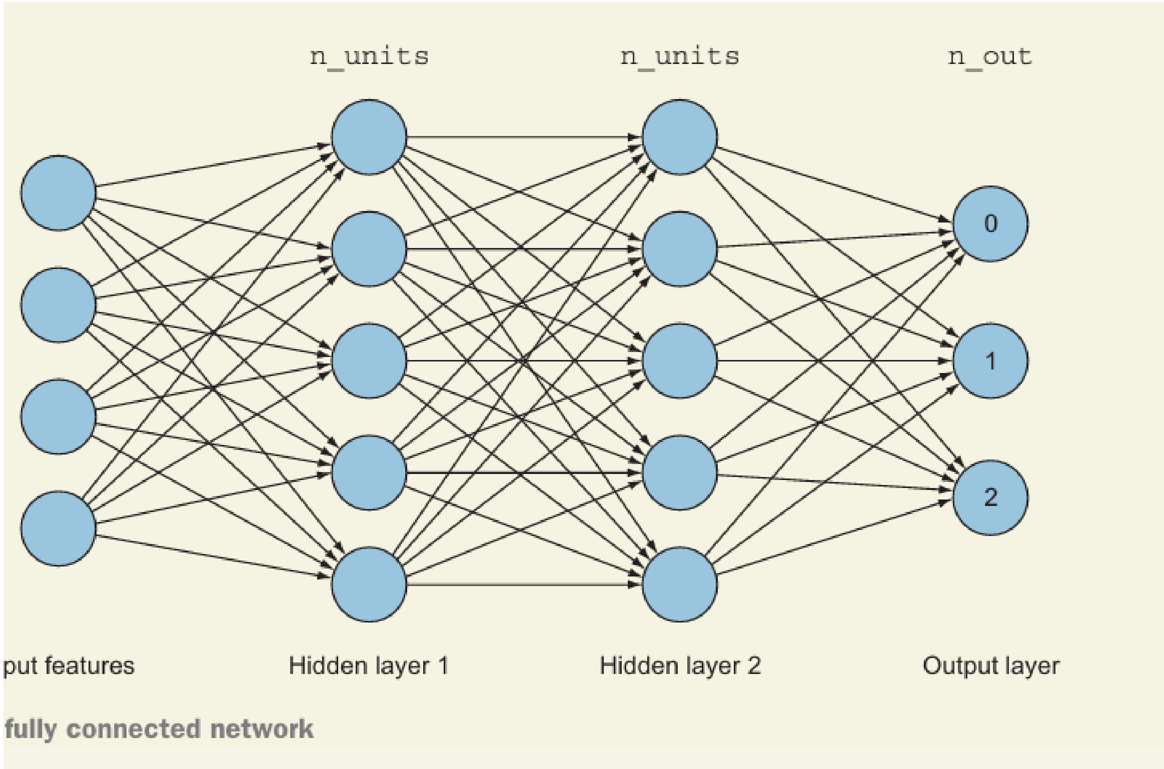


Figure 2.4 Input vectors are fed to the neuron, with weights assigned to represent importance. Calculations performed within the neuron are weighted sum and activation functions.

- A single perceptron is a single artificial neuron.
- It can solve **linearly separable** problems (data that can be split by a straight line).
- **Limitation:** Cannot solve non-linear problems (e.g., XOR problem).

4. Multilayer Perceptron (MLP)



- An MLP consists of multiple perceptrons (neurons) organized in layers.
- It can solve non-linear problems by combining multiple neurons and layers.

5. Why MLP Solves Problems a Single Perceptron Cannot

- **Capturing Non-Linearity:**
  - Single perceptrons can only draw straight lines.
  - MLPs use multiple layers and activation functions to create complex, non-linear decision boundaries.
- **Hidden Layers:**
  - MLPs have **hidden layers** between the input and output layers.
  - Each hidden layer applies transformations to the data, allowing the network to learn complex patterns.
- **Activation Functions:**
  - Non-linear functions (e.g., **ReLU, Sigmoid, Tanh**) introduce non-linearity, enabling the network to model complex relationships.

6. Architecture of an MLP

- **Input Layer:** Receives the input features.
- **Hidden Layers:**
  - Multiple layers of neurons.
  - Each neuron applies a weighted sum and activation function.
- **Output Layer:** Produces the final output (e.g., classification or regression result).

7. How MLP Learns

- **Feedforward:**
  - Input data is passed through the network to produce a prediction.
  - Computes weighted sums and applies activation functions layer by layer.
- **Error Calculation:**
  - Compares the prediction with the actual label using an **error function** (e.g., Mean Squared Error for regression, Cross-Entropy for classification).
- **Backpropagation:**
  - Propagates the error backward through the network.
  - Updates the weights using **gradient descent** to minimize the error.

8. Key Hyperparameters in Neural Networks

- **Number of Hidden Layers:**
  - More layers = more capacity to learn complex patterns.
  - Too many layers can lead to **overfitting** (memorizing training data instead of generalizing).
  - Start with a small network and gradually increase layers.
- **Activation Functions:**
  - **ReLU (Rectified Linear Unit):** Commonly used in hidden layers.
  - **Softmax:** Used in the output layer for classification problems.
- **Error Function:**
  - **Mean Squared Error (MSE):** For regression problems.
  - **Cross-Entropy:** For classification problems.
- **Optimizer:**
  - Algorithms like **Gradient Descent, Adam, and RMSprop** are used to update weights.
  - **Adam** is a popular choice for its efficiency.

- **Batch Size:**
  - Number of samples processed before updating weights.
  - Common values: **32, 64, 128, 256**.
  - Larger batch sizes = faster learning but require more memory.
- **Number of Epochs:**
  - One epoch = one full pass through the training data.
  - Increase epochs until validation accuracy stops improving (to avoid overfitting).
- **Learning Rate:**
  - Controls the step size during weight updates.
  - Start with a default value (e.g., **0.01**) and adjust as needed.

9. How MLP Captures Non-Linearity

- **Combination of Layers:**
  - Each layer applies a transformation to the data, allowing the network to learn hierarchical features.
- **Activation Functions:**
  - Non-linear functions like **ReLU** introduce non-linearity, enabling the network to model complex relationships.
- **Hidden Layers:**
  - Multiple hidden layers allow the network to learn increasingly abstract features.

10. Practical Tips

- **Start Small:**
  - Begin with a small network (e.g., **1-2 hidden layers**) and gradually increase complexity.
- **Use ReLU in Hidden Layers:**
  - **ReLU** is simple and effective for most problems.
- **Monitor Overfitting:**
  - Use techniques like **dropout** or **regularization** to prevent overfitting.
- **Tune Hyperparameters:**
  - Experiment with **learning rate, batch size, and number of epochs** to find the best configuration.

11. Summary

- **Single Perceptron:** Can only solve **linearly separable** problems.
- **MLP:** Can solve **non-linear** problems by combining multiple layers and neurons.
- **Key Components:**
  - **Input layer, hidden layers, output layer**
  - **Weighted sums and activation functions**
- **Learning Process:**
  - **Feedforward, error calculation, backpropagation**
- **Hyperparameters:**
  - **Number of layers, activation functions, batch size, learning rate, etc.**