Deep Learning

The Artificial Neuron

- Modeled after the human brain's neurons
- The basic building block of any neural network or deep learning model
- We will quickly go over the evolution of the artificial neuron
- Note: Everything discussed in these slides is from the software perspective (it does not discuss hardware of data)

The M-P Neuron

- Considered by many the first artificial neuron

- The McCulloch-Pitts Neuron (a.k.a. the MCP Neuron); created by Warren McCulloch and Walter Pitts in 1943

- Based on the concept that the brain’s decision-making process can be modeled using boolean functions

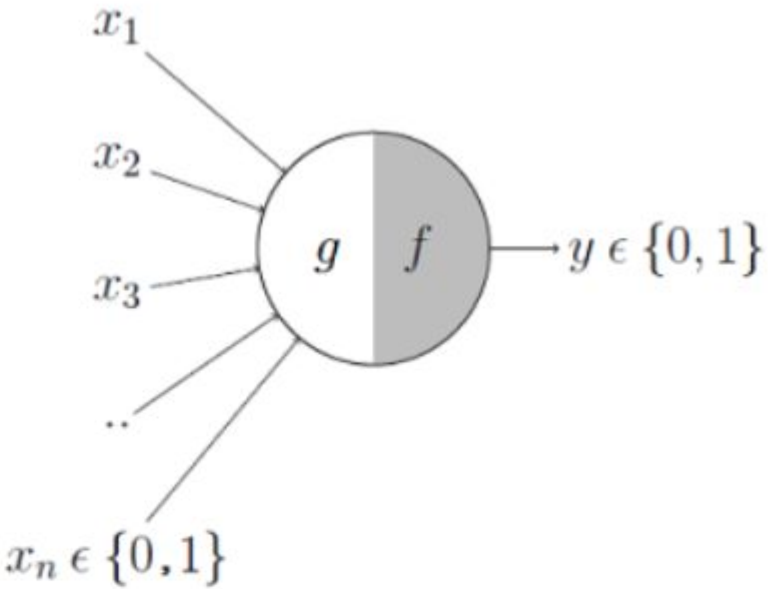
The M-P Neuron

- Contains 4 parts:
  - a set of inputs **x** where each input can be **0** or **1** (i.e., **false** or **true**, respectively), can be **excitatory** or **inhibitory**
  - a summing function **g** to sum **x**
  - a decision (or piecewise) function **f**
  - **f** also contains a threshold value, **θ** (**theta**)

- A neuron outputs **0** or **1** (i.e., **false** or **true**, respectively)

- The real power of neurons is to use them in sequence (i.e., in a network)

The M-P Neuron



$$g(x_1, x_2, x_3, ..., x_n) = g(\mathbf{x}) = \sum_{i=1}^{n} x_i$$

$$\begin{aligned} y = f(g(\mathbf{x})) &= 1 \quad if \quad g(\mathbf{x}) \geq \theta \\ &= 0 \quad if \quad g(\mathbf{x}) < \theta \end{aligned}$$

The M-P Neuron: Limitations

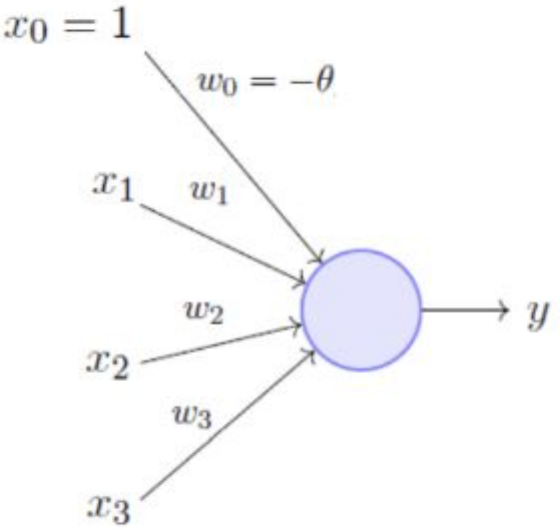
- All input is equivalent in importance
- Cannot model non-boolean data (e.g., cannot process images, text, etc.)
- Must identify and hard-code every input
- Must find/calculate and hard-code every threshold
- Individual neurons can only handle linearly separable decisions (i.e., the decision space must be separated into 2 parts, as **true** or **false**, **1** or **0**)
    - Can still model some non-linearly separable decisions, but must use multiple neurons to do it

The Perceptron

- First proposed by Frank Rosenblatt in 1958 and further refined by Marvin Minsky and Seymour Papert in 1969

- Used the M-P Neuron as a foundation, but with 3 main differences:
  - Input (**x**) are now real-valued numbers; **weights** have been included for each input (**x**), they are to be multiplied with each of their corresponding input (**x**) before being summed by **g**
  - A mechanism (i.e. **learning algorithm**) to learn/determine the best/optimal value for those weights has been introduced; example data is needed for the algorithm to learn
  - **θ** (**theta**), has been removed as a threshold value and reintroduced as a **bias** term (i.e., **θ** (**theta**) can be learned like the weights and doesn’t need to be determined in advance)

The Perceptron



$$f = 1 \quad if \sum_{i=1}^{n} w_i * x_i \geq \theta$$
$$= 0 \quad if \sum_{i=1}^{n} w_i * x_i < \theta$$

⇒

$$f = 1 \quad if \sum_{i=1}^{n} w_i * x_i - \theta \geq 0$$
$$= 0 \quad if \sum_{i=1}^{n} w_i * x_i - \theta < 0$$

⇒

$$f = 1 \quad if \sum_{i=0}^{n} w_i * x_i \geq 0$$
$$= 0 \quad if \sum_{i=0}^{n} w_i * x_i < 0$$
$$where, \quad x_0 = 1 \quad and \quad w_0 = -\theta$$

The Perceptron: Learning Algorithm

- Note: the learning algorithm is completely separate from the Perceptron (i.e., you can replace the current learning algorithm with a different learning algorithm and the perceptron will not change)

- Is proven to converge every time, for any given input

Algorithm: Perceptron Learning Algorithm

$P \leftarrow inputs \quad with \quad label \quad 1;$
$N \leftarrow inputs \quad with \quad label \quad 0;$
Initialize $\mathbf{w}$ randomly;
while $!convergence$ **do**
  Pick random $\mathbf{x} \in P \cup N$;
  if $\mathbf{x} \in P \quad and \quad \mathbf{w}.\mathbf{x} < 0$ **then**
    $\mathbf{w} = \mathbf{w} + \mathbf{x}$;
  end
  if $\mathbf{x} \in N \quad and \quad \mathbf{w}.\mathbf{x} \geq 0$ **then**
    $\mathbf{w} = \mathbf{w} - \mathbf{x}$;
  end
end
//the algorithm converges when all the inputs are classified correctly

The Perceptron: Limitations

- Old limitations that are still applicable to the Perceptron:
  - Must identify and hard-code every input
  - Individual neurons can only handle linearly separable decisions

- The learning algorithm only converges based on the example input given (i.e., given a new example, it cannot be guaranteed that the neuron will produce the correct result)

The Sigmoid Neuron

- Came about from the culmination of a number of papers and researchers in the mid 1980s

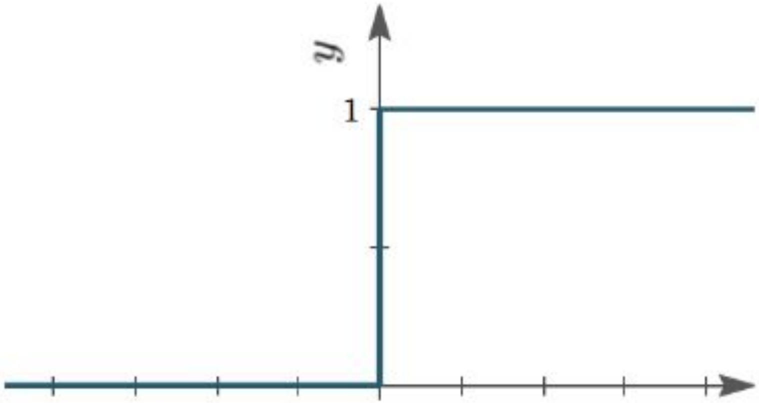
- Considered the “normal” or “average” neuron using in neural networks today

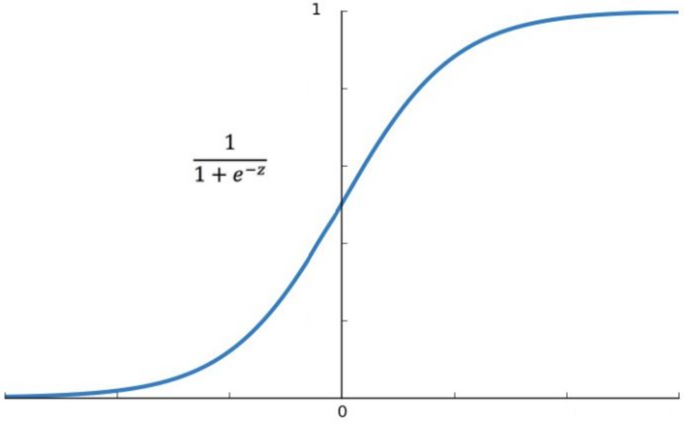
- Uses the Perceptron as a base, but with 1 major difference
    - **f** has been modified from the (piecewise) step function to the sigmoid function (a.k.a. the **activation function**); instead of producing **0** or **1**, it produces a real-valued number between 0 and 1

- **f** can use other **activation functions** (e.g., tanh, ReLU, etc.)

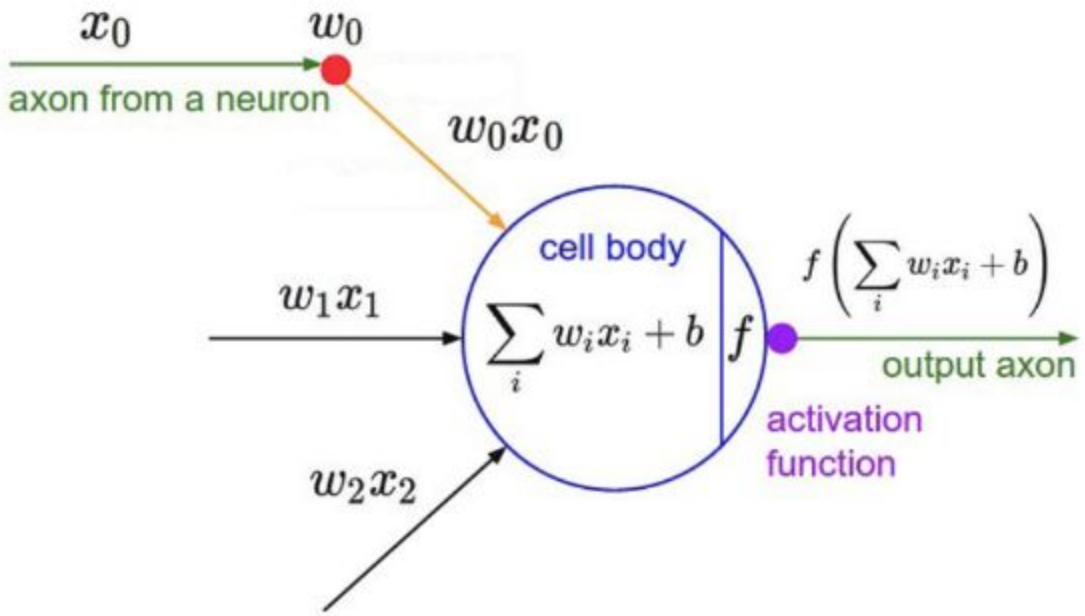
- Must use a more sophisticated learning algorithm (e.g., Mean Squared Error, Gradient Descent, etc.)

The Sigmoid Neuron





The Sigmoid Neuron



The Sigmoid Neuron: Limitations

- Complete convergence is often no longer possible

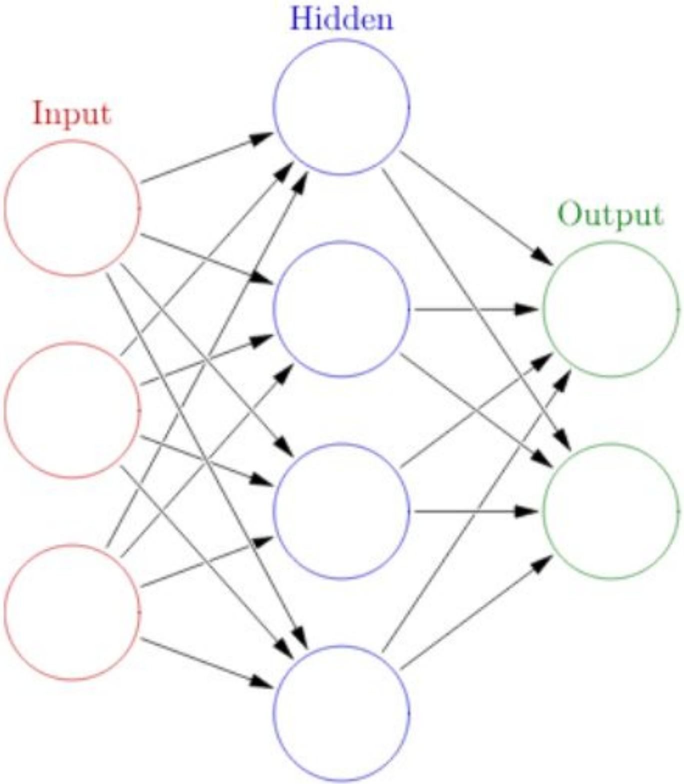
Neural Networks

- Neurons are most useful connected together in a network

- We will look at the most basic neural network (and the foundation of all modern neural networks and deep learning research), the Feed-Forward Neural Network (FFNN)

- The FFNN has 3 main layers:
    - The **Input Layer** - the input data to the network; can be any real-valued data (e.g., pixel values in an image, sales data, population data, etc.)
    - The **Hidden Layer** - a set of artificial neurons that are densely connected they transform input based on their summing and activation functions and pass their output to subsequent layers
    - The **Output Layer** - a final data transformation to the proper output format, the final transformation is dependent on the type of output (e.g., if it is a categorization problem it will likely use a softmax function)

Neural Networks



Neural Networks

- There can be multiple levels of hidden layers (this is what makes the learning "deep")

- Note: We can no longer tell what each artificial neuron represents