Neurons

- Our brain contains about 86 billion neurons.
- Each neuron receives signals from other neurons via its many dendrites (input).
- Each neuron has a single axon (output).
- Neuron make on average 7,000 synaptic connections.
- Signals are sent via an electrochemical process.
- When a neuron fires, it starts a chain reaction that propagates information.
- There are excitatory and inhibitory synapses.

See Benman (2013) for more details.

Neural networks

Single neuron model:

Input: \( x_1, x_2, x_3 \) (and +1 intercept).
Output: \( h_{W,b}(x) = f(W^T x) = f(W_1 x_1 + W_2 x_2 + W_3 x_3 + b) \),
where \( f \) is the sigmoid function:

\[
f(x) = \frac{1}{1 + e^{-x}}.
\]

Other common choice for \( f \):

\[
f(x) = \tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}.
\]
Neural networks (cont.)

The function \( f \) acts as an activation function.

Idea: Depending on the input of the neuron and the strength of the links, the neuron “fires” or not.

Neural network models

A neural networks model is obtained by hooking together many neurons so that the output of one neuron becomes the input of another neuron.

Note: Each layer includes an intercept “+1” (or bias unit)
- Leftmost layer = input layer.
- Rightmost layer = output layer.
- Middle layers = hidden layers (not observed).

We will let \( n_l \) denote the number of layers in our model (\( n_l = 3 \) in the above example).

Notation

- \( n_l \) = number of layers.
- We denote the layers by \( L_1, \ldots, L_{n_l} \), so \( L_1 \) = input layer and \( L_{n_l} \) = output layer.
- \( W(l)_{ij} \) = weight associated with the connection between unit \( j \) in layer \( l \), and unit \( i \) in layer \( l+1 \). (Note the order of the indices.)
- \( b(l)_i \) = bias associated with unit \( i \) in layer \( l+1 \).

In above example: \((W, b) = (W^{(1)}, b^{(1)}, W^{(2)}, b^{(2)})\). Here \( W^{(1)} \in \mathbb{R}^{3 \times 3}, W^{(2)} \in \mathbb{R}^{1 \times 3}, b^{(1)} \in \mathbb{R}^3, b^{(2)} \in \mathbb{R} \).

Activation

- We denote by \( a^{(l)}_i \) the activation of unit \( i \) in layer \( l \).
- We let \( a^{(1)}_1 = x_i \) (input).

We have:
\[
\begin{align*}
a^{(2)}_1 &= f(W^{(1)}_{11} x_1 + W^{(1)}_{12} x_2 + W^{(1)}_{13} x_3 + b^{(1)}_1) \\
a^{(2)}_2 &= f(W^{(1)}_{21} x_1 + W^{(1)}_{22} x_2 + W^{(1)}_{23} x_3 + b^{(1)}_2) \\
a^{(2)}_3 &= f(W^{(1)}_{31} x_1 + W^{(1)}_{32} x_2 + W^{(1)}_{33} x_3 + b^{(1)}_3) \\
h_{W,b} &= a^{(3)}_1 = f(W^{(2)}_{11} a^{(2)}_1 + W^{(2)}_{12} a^{(2)}_2 + W^{(2)}_{13} a^{(2)}_3 + b^{(2)}_1).
\end{align*}
\]
Compact notation

In what follows, we will let $z_i^{(l)}$ be the total weighted sum of inputs to unit $i$ in layer $l$ (including the bias term):

$$z_i^{(l)} := \sum_j W_{ij}^{(l-1)} a_j^{(l-1)} + b_i^{(l-1)} \quad (l \geq 2).$$

- Note that $a_i^{(0)} = f(z_i^{(0)})$.
- For example:
  $$z_i^{(2)} = \sum_{j=1}^{3} W_{ij}^{(1)} x_j + b_i^{(1)} \quad i = 1, 2, 3.$$

We extend $f$ elementwise: $f([v_1, v_2, v_3]) = [f(v_1), f(v_2), f(v_3)]$.

Using the above notation, we have:

- $z^{(2)} = W^{(1)} x + b^{(1)}$
- $a^{(2)} = f(z^{(2)})$
- $z^{(3)} = W^{(2)} a^{(2)} + b^{(2)}$
- $h_{W,b} = a^{(3)} = f(z^{(3)})$.

Forward propagation

The previous process is called the **forward propagation step**.

- Recall that we defined $a^{(1)} = x$ (the input).
- The forward propagation can therefore be written as:
  $$z^{(l+1)} = W^{(l)} a^{(l)} + b^{(l)}$$
  $$a^{(l+1)} = f(z^{(l+1)}).$$

Using matrix-vector operations, we can take advantage of fast linear algebra routines to quickly perform calculations in our network.

- Can use different architectures (i.e., patterns of connectivity between neurons).
- Typically, we use multiple densely connected layers.
- In that case, we obtain a feedforward neural network (no directed loops or cycles).

Multiple outputs

Neural networks may also have multiple outputs:

![Diagram of a neural network with multiple outputs](source: UFLDL tutorial)

- To train this network, we need observations $(x^{(i)}, y^{(i)})$ with $y^{(i)} \in \mathbb{R}^2$.
- Useful for applications where the output is multivariate (e.g., medical diagnosis application where output is whether or not a patient has a list of diseases).
- Useful to encode or compress information.

The universal approximation theorem

- What kind of functions can we approximate with neural networks?
- The following result shows that neural networks have a "universal" approximation property.

**Theorem:** (Cybenko, 1989) A single layer feedforward neural network can uniformly approximate any continuous function defined on a compact subset $K$ of $\mathbb{R}^n$.

- A subset of $\mathbb{R}^n$ is **compact** if it is closed $(x_n \in K$ and $x_n \to x$ implies $x \in K)$ and bounded ($||x|| \leq C$ for all $x \in K$).

**Example:** Let $f$ be any continuous function defined on the unit cube $[0, 1]^3$. Then for every $\epsilon > 0$, there exists a feedforward neural network $f_{W,b}$ with one layer such that

$$|f(x) - f_{W,b}(x)| < \epsilon \quad \forall x \in [0, 1]^3.$$