MATH 567: Introduction to Data Mining and Analysis
Introduction to Neural networks

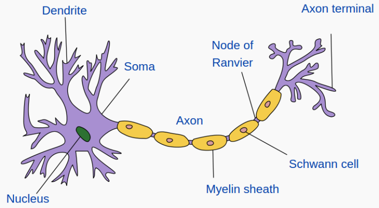
Dominique Guillot

Departments of Mathematical Sciences
University of Delaware

April 10, 2017

This lecture is based on the UFLDL tutorial (http://deeplearning.stanford.edu/)

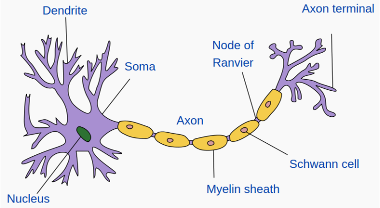
Neurons



Neuron representation (Source: Wiki).

- Our brain contains about 86 billion neurons.

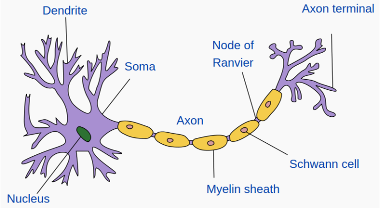
Neurons



Neuron representation (Source: Wiki).

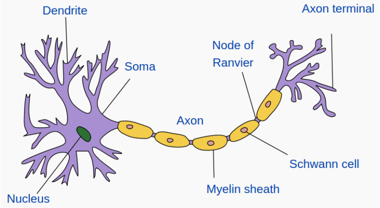
- Our brain contains about 86 billion neurons.
- Each neuron receives signals from other neurons via its many dendrites (input).

Neurons



Neuron representation (Source: Wiki).

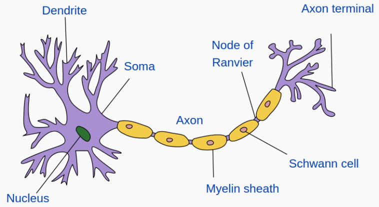
- Our brain contains about 86 billion neurons.
- Each neuron receives signals from other neurons via its many dendrites (input).
- Each neuron has a single axon (output).



Neuron representation (Source: Wiki).

- 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.

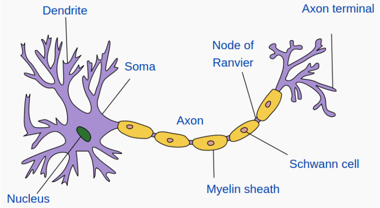
Neurons



Neuron representation (Source: Wiki).

- 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.

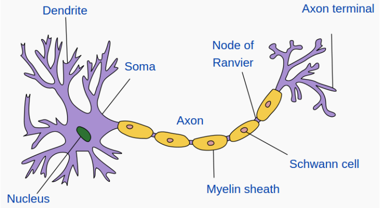
Neurons



Neuron representation (Source: Wiki).

- 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.

Neurons



Neuron representation (Source: Wiki).

- 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 Izenman (2013) for more details.

Neurons (cont.)

- Our brain *learns* by changing the **strengths** of the connections between neurons or by **adding** or **removing** such connections.

Neurons (cont.)

- Our brain *learns* by changing the **strengths** of the connections between neurons or by **adding** or **removing** such connections.
- Relating brain networks to *functions* is still a very challenging problem.

Neurons (cont.)

- Our brain *learns* by changing the **strengths** of the connections between neurons or by **adding** or **removing** such connections.
- Relating brain networks to *functions* is still a very challenging problem.
- Constructing a "universal" learning machine/algorithm?

Neurons (cont.)

- Our brain *learns* by changing the **strengths** of the connections between neurons or by **adding** or **removing** such connections.
- Relating brain networks to *functions* is still a very challenging problem.
- Constructing a "universal" learning machine/algorithm?
- In some sense, no machine learning algorithm is universally better than any other (Goodfellow et al., "Deep learning"):

Neurons (cont.)

- Our brain *learns* by changing the **strengths** of the connections between neurons or by **adding** or **removing** such connections.
- Relating brain networks to *functions* is still a very challenging problem.
- Constructing a "universal" learning machine/algorithm?
- In some sense, no machine learning algorithm is universally better than any other (Goodfellow et al., "Deep learning"):

Theorem: (No free lunch theorem) (Wolpert, 1996) Averaged over all possible data generating distributions, every classification algorithm has the same error rate when classifying previously unobserved points.

Neurons (cont.)

- Our brain *learns* by changing the **strengths** of the connections between neurons or by **adding** or **removing** such connections.
- Relating brain networks to *functions* is still a very challenging problem.
- Constructing a "universal" learning machine/algorithm?
- In some sense, no machine learning algorithm is universally better than any other (Goodfellow et al., "Deep learning"):

Theorem: (No free lunch theorem) (Wolpert, 1996) Averaged over all possible data generating distributions, every classification algorithm has the same error rate when classifying previously unobserved points.

- Still, there is hope to construct an *algorithm* that performs well at many tasks (e.g. the human brain).

Neurons (cont.)

- Our brain *learns* by changing the **strengths** of the connections between neurons or by **adding** or **removing** such connections.
- Relating brain networks to *functions* is still a very challenging problem.
- Constructing a “universal” learning machine/algorithm?
- In some sense, no machine learning algorithm is universally better than any other (Goodfellow et al., “Deep learning”):

Theorem: (No free lunch theorem) (Wolpert, 1996) Averaged over all possible data generating distributions, every classification algorithm has the same error rate when classifying previously unobserved points.

- Still, there is hope to construct an *algorithm* that performs well at many tasks (e.g. the human brain).

Neural networks:

- **Inspired** by neuroscience (probably very far from real neurons).

Neurons (cont.)

- Our brain *learns* by changing the **strengths** of the connections between neurons or by **adding** or **removing** such connections.
- Relating brain networks to *functions* is still a very challenging problem.
- Constructing a "universal" learning machine/algorithm?
- In some sense, no machine learning algorithm is universally better than any other (Goodfellow et al., "Deep learning"):

Theorem: (No free lunch theorem) (Wolpert, 1996) Averaged over all possible data generating distributions, every classification algorithm has the same error rate when classifying previously unobserved points.

- Still, there is hope to construct an *algorithm* that performs well at many tasks (e.g. the human brain).

Neural networks:

- **Inspired** by neuroscience (probably very far from real neurons).
- Use multiple layers of neurons to represent data.

Neurons (cont.)

- Our brain *learns* by changing the **strengths** of the connections between neurons or by **adding** or **removing** such connections.
- Relating brain networks to *functions* is still a very challenging problem.
- Constructing a "universal" learning machine/algorithm?
- In some sense, no machine learning algorithm is universally better than any other (Goodfellow et al., "Deep learning"):

Theorem: (No free lunch theorem) (Wolpert, 1996) Averaged over all possible data generating distributions, every classification algorithm has the same error rate when classifying previously unobserved points.

- Still, there is hope to construct an *algorithm* that performs well at many tasks (e.g. the human brain).

Neural networks:

- **Inspired** by neuroscience (probably very far from real neurons).
- Use multiple layers of neurons to represent data.
- Very popular in computer vision, natural language processing, and many other fields.

Neurons (cont.)

- Our brain *learns* by changing the **strengths** of the connections between neurons or by **adding** or **removing** such connections.
- Relating brain networks to *functions* is still a very challenging problem.
- Constructing a "universal" learning machine/algorithm?
- In some sense, no machine learning algorithm is universally better than any other (Goodfellow et al., "Deep learning"):

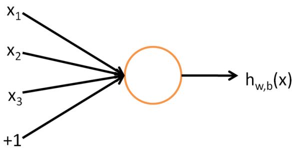
Theorem: (No free lunch theorem) (Wolpert, 1996) Averaged over all possible data generating distributions, every classification algorithm has the same error rate when classifying previously unobserved points.

- Still, there is hope to construct an *algorithm* that performs well at many tasks (e.g. the human brain).

Neural networks:

- **Inspired** by neuroscience (probably very far from real neurons).
- Use multiple layers of neurons to represent data.
- Very popular in computer vision, natural language processing, and many other fields.
- Today, neural network models are often called *deep learning*.

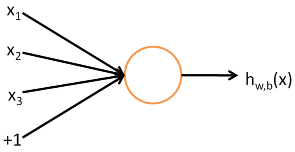
Single neuron model:



Source: UFLDL Tutorial

Neural networks

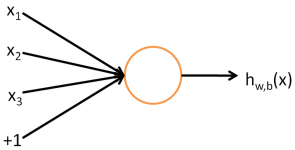
Single neuron model:



Source: UFLDL Tutorial

Input: x_1, x_2, x_3 (and $+1$ intercept).

Single neuron model:

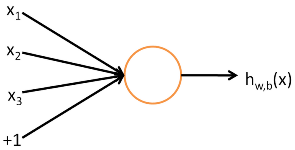


Source: UFLDL Tutorial

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:

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

Single neuron model:



Source: UFLDL Tutorial

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:

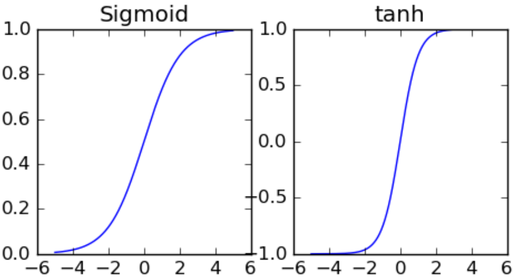
$$f(x) = \frac{1}{1 + e^{-x}}.$$

Other common choice for f:

$$f(x) = \tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}.$$

Neural networks (cont.)

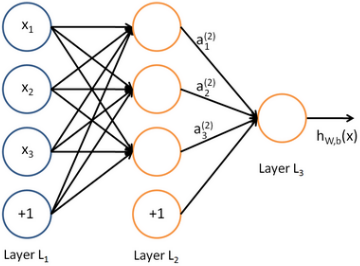
The function f acts as an **activation** function.



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

Neural network models

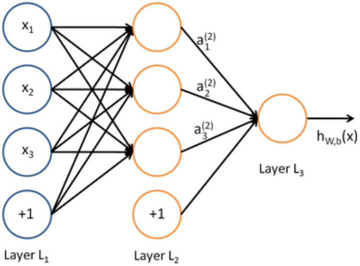
A **neural networks model** is obtained by hooking together many neurons so that the output of one neuron becomes the input of another neuron.



Source: UFLDL tutorial

Neural network models

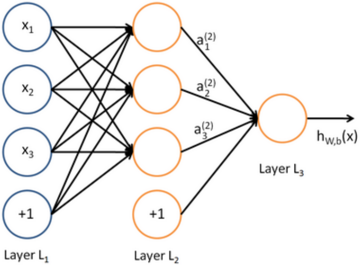
A **neural networks model** is obtained by hooking together many neurons so that the output of one neuron becomes the input of another neuron.



Source: UFLDL tutorial

Note: Each layer includes an intercept "$+1$" (or bias unit)

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



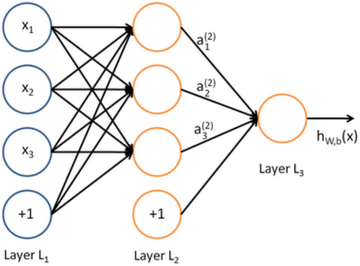
Source: UFLDL tutorial

Note: Each layer includes an intercept "$+1$" (or bias unit)

- Leftmost layer = input layer.
- Rightmost layer = output layer.
- Middle layers = hidden layers (not observed).

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.



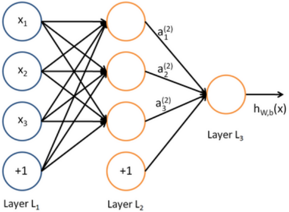
Source: UFLDL tutorial

Note: Each layer includes an intercept "$+1$" (or bias unit)

- Leftmost layer = input layer.
- Rightmost layer = output layer.
- Middle layers = hidden layers (not observed).

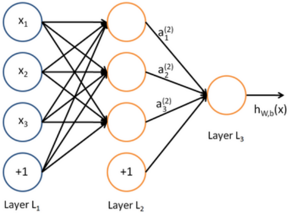
We will let n_l denote the **number of layers** in our model ($n_l = 3$ in the above example).

Notation



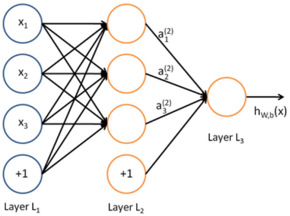
- n_l = number of layers.

Notation



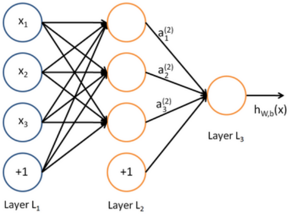
- n_l = number of layers.
- We denote the layers by $L_1, \ldots, L_{n_l}$, so L_1 = input layer and L_{n_l} = output layer.

Notation



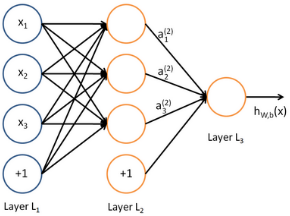
- n_l = number of layers.
- We denote the layers by $L_1, \ldots, L_{n_l}$, so L_1 = input layer and L_{n_l} = output layer.
- $W_{ij}^{(l)}$ = weight associated with the connection between unit j in layer l, and unit i in layer $l+1$. (Note the order of the indices.)

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.
- $W_{ij}^{(l)}$ = weight associated with the connection between unit j in layer l, and unit i in layer $l+1$. (Note the order of the indices.)
- $b_i^{(l)}$is the bias associated with unit i in layer $l+1$.

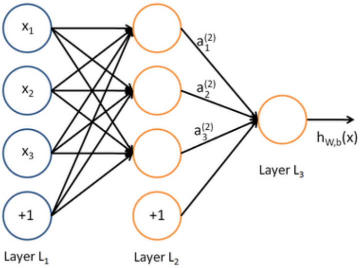
Notation



- 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.
- $W_{ij}^{(l)}$ = weight associated with the connection between unit j in layer l, and unit i in layer $l+1$. (Note the order of the indices.)
- $b_i^{(l)}$is the bias associated with unit i in layer $l+1$.

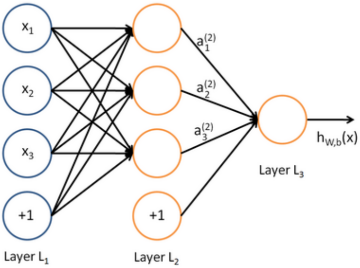
In above example: $(W, b) = (W^{(1)}, b^{(1)}, W^{(2)}, b^{(2)})$. Here $W^{(1)} \in \mathbb{R}^{3\times3}$, $W^{(2)} \in \mathbb{R}^{1\times3}$, $b^{(1)} \in \mathbb{R}^3$, $b^{(2)} \in \mathbb{R}$.

Activation



- We denote by $a_i^{(l)}$ the **activation** of unit i in layer l.
- We let $a_i^{(1)} = x_i$ (input).

Activation



- We denote by $a_i^{(l)}$ the **activation** of unit i in layer l.
- We let $a_i^{(1)} = x_i$ (input).

We have:

$$\begin{aligned}
a_1^{(2)} &= f(W_{11}^{(1)}x_1 + W_{12}^{(1)}x_2 + W_{13}^{(1)}x_3 + b_1^{(1)}) \\
a_2^{(2)} &= f(W_{21}^{(1)}x_1 + W_{22}^{(1)}x_2 + W_{23}^{(1)}x_3 + b_2^{(1)}) \\
a_3^{(2)} &= f(W_{31}^{(1)}x_1 + W_{32}^{(1)}x_2 + W_{33}^{(1)}x_3 + b_3^{(1)}) \\
h_{W,b} = a_1^{(3)} &= f(W_{11}^{(2)}a_1^{(2)} + W_{12}^{(2)}a_2^{(2)} + W_{13}^{(2)}a_3^{(2)} + b_1^{(2)}).
\end{aligned}$$

Compact notation

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

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

Compact notation

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

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

- Note that that $a_i^{(l)} = f(z_i^{(l)})$.

Compact notation

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

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

- Note that that $a_i^{(l)} = f(z_i^{(l)})$.
- For example:

$$z_i^{(2)} = \sum_{j=1}^{3} W_{ij}^{(1)} x_j + b_i^{(1)} \qquad i = 1, 2, 3.$$

Compact notation

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

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

- Note that that $a_i^{(l)} = f(z_i^{(l)})$.
- For example:

$$z_i^{(2)} = \sum_{j=1}^{3} W_{ij}^{(1)} x_j + b_i^{(1)} \qquad i = 1, 2, 3.$$

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

Compact notation

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

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

- Note that that $a_i^{(l)} = f(z_i^{(l)})$.
- For example:

$$z_i^{(2)} = \sum_{j=1}^{3} W_{ij}^{(1)} x_j + b_i^{(1)} \qquad i = 1, 2, 3.$$

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:

$$\begin{aligned} z^{(2)} &= W^{(1)} x + b^{(1)} \\ a^{(2)} &= f(z^{(2)}) \\ z^{(3)} &= W^{(2)} a^{(2)} + b^{(2)} \\ h_{W,b} &= a^{(3)} = f(z^{(3)}). \end{aligned}$$

Forward propagation

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

Forward propagation

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

- Recall that we defined $a^{(1)} = x$ (the input).

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:

$$z^{(l+1)} = W^{(l)}a^{(l)} + b^{(l)}$$
$$a^{(l+1)} = f(z^{(l+1)}).$$

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.

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., pattens of connectivity between neurons).

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., pattens of connectivity between neurons).
- Typically, we use multiple densely connected layers.

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:

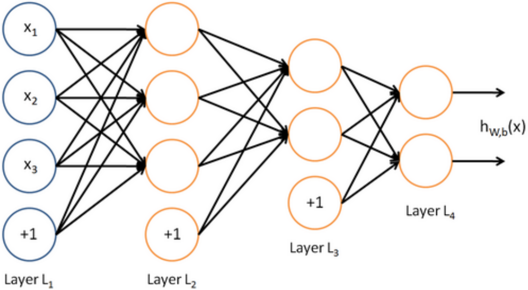
$$
\begin{aligned}
z^{(l+1)} &= W^{(l)} a^{(l)} + b^{(l)} \\
a^{(l+1)} &= f(z^{(l+1)}).
\end{aligned}
$$

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., pattens 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

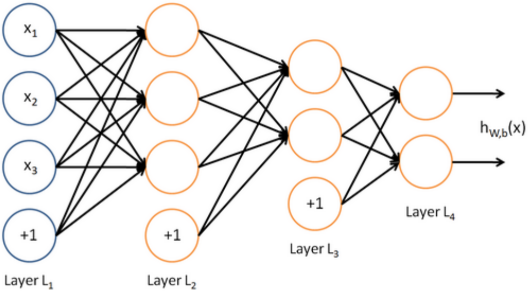
Neural networks may also have multiple outputs:



Source: UFLDL tutorial

Multiple outputs

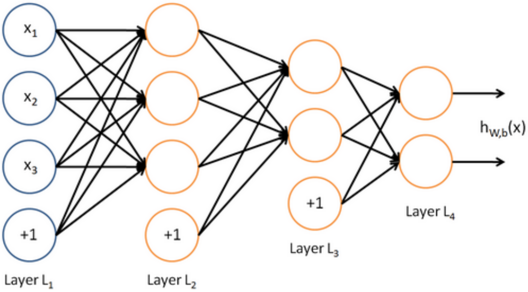
Neural networks may also have multiple outputs:



Source: UFLDL tutorial

- To train this network, we need observations $(x^{(i)}, y^{(i)})$ with $y^{(i)} \in \mathbb{R}^2$.

Neural networks may also have multiple outputs:

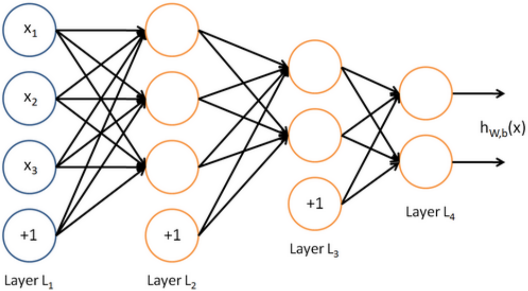


Source: UFLDL tutorial

- 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).

Multiple outputs

Neural networks may also have 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 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.

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 $\mathbb{R}^n$.

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 $x \in K$).

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

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