LING3804: Lecture Notes 1
From Artificial Intelligence to Artificial Neurons

AI large language models like ChatGPT (Generative Pre-trained Transformer) are quite influential.

This course will explore similarities and differences between human and AI models of language.

We start with artificial neurons and neural networks, then scale up to modern large language models.

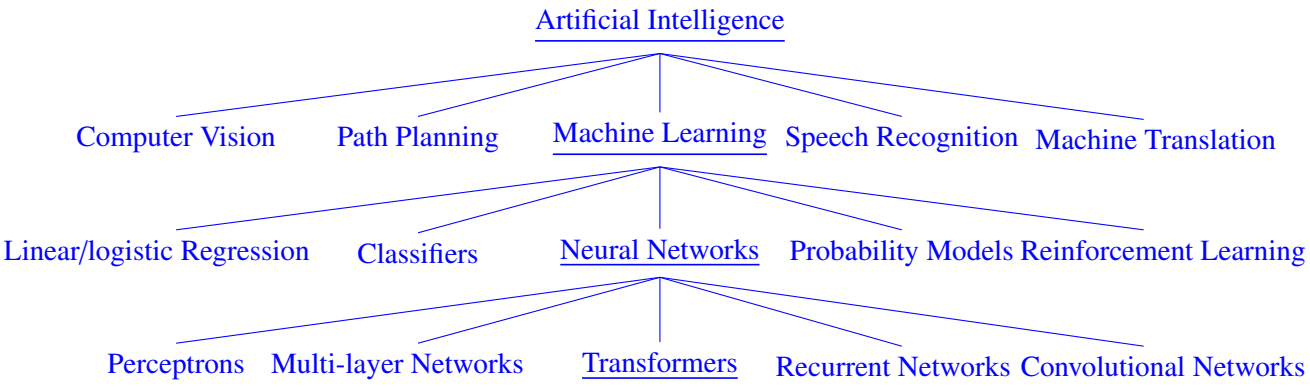
These are mostly mathematical models, but this course will introduce all the necessary concepts.

No programming is required in this course.

Contents

1.1 Artificial Intelligence, Machine Learning and Neural Networks

The field of Artificial Intelligence looks like this, from general areas (top) to specific areas (bottom):



The field of Artificial Intelligence 'AI' is very broad, contains the following nested areas:

1. **Artificial Intelligence** is teaching computers to do things people do without being taught: computer vision, path planning, speech recognition, machine translation, machine learning
2. **Machine Learning** is teaching computers to do things by induction, from examples: linear/logistic regression, probability models, reinforcement learning, neural networks

3. **Neural Networks** are systems that learn via computational models of biological neurons: perceptrons, multi-layer networks, recurrent networks, convolutional networks, transformers
4. **Large Language Models** are models trained on lots of text data, e.g. 1,000,000,000+ words: n-gram models, long short-term memory, transformers (ChatGPT, etc., get called 'AI' now)

So transformers are neural network models that learn from large amounts of language data.

We'll start with neuron models, then show how they learn, then scale up to transformers.

1.2 Biology of neural activation

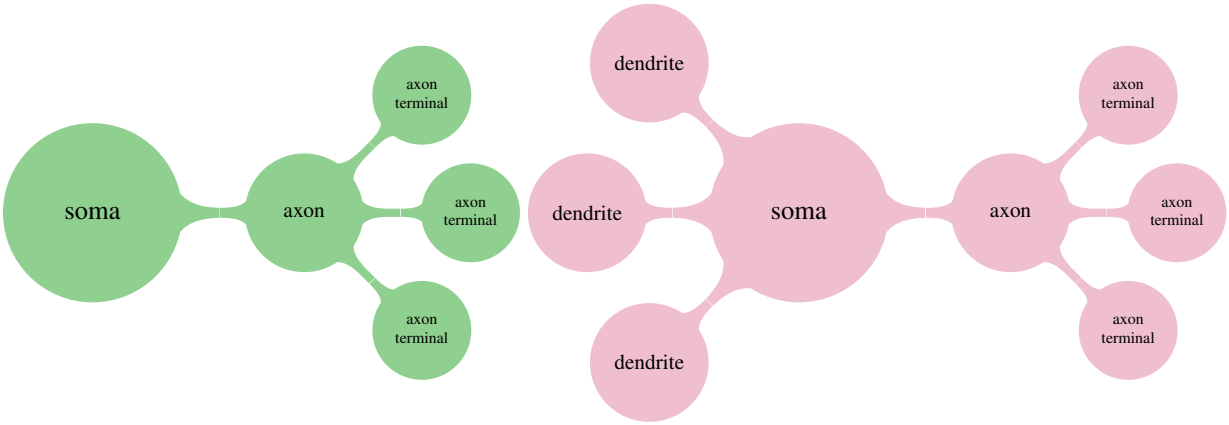
Artificial neural units used in AI neural networks are based on biological neurons.

Biological neurons look like trees, with roots, trunks, and branches. A neuron has:

- **dendrites:** 'roots' near other neurons to receive chemical signals
- an **axon:** a 'trunk' along which the neuron propagates electric potential
- **axon terminals:** 'branches' near other neurons to send chemical signals

It also has:

- **synapses:** gaps between terminals and dendrites that permit thresholding
- **neurotransmitters:** chemicals that carry signals across synapses
- **vesicles:** bubbles in axon terminals that contain neurotransmitters
- **receptors:** attachment sites for neurotransmitters on dendrites



Neurons transmit signals or 'fire' by suddenly changing electric potential:

1. start with more K^+ but much fewer Na^+ ions than outside, creating **membrane potential**;
2. (dendrites) receptors receive neurotransmitters, open **ligand-gated** channels;

3. (dendrites) ligand-gated channels let Ca^{++}/Cl^{-} in or K^{+} out, changing potential
   (this is a **linear** function on the sum of positive/negative ions in the neuron);
4. (axon) if potential changes enough, **voltage-gated** channels come open;
5. (axon) voltage-gated channels let in many Na^{+}/Ca^{++} ions; neuron **depolarizes**
   (this is a non-linear **threshold** function on the sum of positive/negative ions in the neuron);
6. (axon terminals) depolarization allows **vesicles** to meet surface, release neurotransmitters;
7. depolarization makes voltage-gated channels let out K^{+}, **repolarize** cell;
8. ion pumps on surface put back Ca^{++},Cl^{-},Na^{+},K^{+}, neurotransmitters.

Synaptic connections may be **positive or negative**, e.g.:

1. **pyramidal neurons** may emit neurotransmitters that gate in positive ions
2. **interneurons** may emit neurotransmitters that gate in negative ions

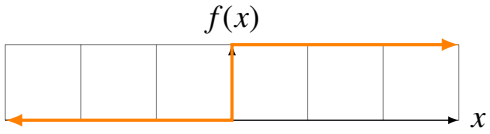
Synaptic connections also have **weights**:

1. repeated firing removes Mg blockers, so the 'rest state' depolarizes a bit
2. fewer Mg blockers increases phosphate, makes receptors more efficient
3. fewer Mg blockers triggers construction of more receptors (to let in more ions)

1.3 A simple model of neural activation [Mcculloch & Pitts, 1943]

The **linear function** and **threshold function** can be modeled mathematically:

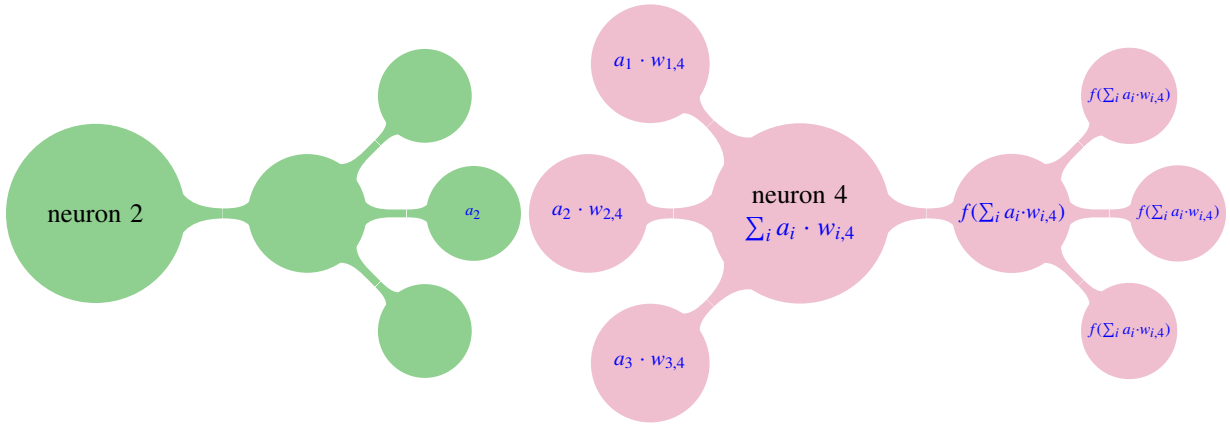
1. given a_i: real-valued activation of artificial neural units i,
2. given $w_{i,j}$: real-valued synaptic weight (positive/negative) of connection from unit i to unit j,
3. given $\sum_i a_i \cdot w_{i,j}$: connection-weighted (**linear**) sum of impinging neural units
   (the capital sigma $\sum_i$ denotes a sum over all values of its subscript i: $a_1 \cdot w_{1,4} + a_2 \cdot w_{2,4} + \ldots$),
4. given f: **threshold** function, e.g. Heaviside/step $f(x) = \begin{cases} 1 & \text{if } x \geq 0 \\ 0 & \text{if } x < 0 \end{cases}$,



the activation value a_j of a unit with antecedent unit activations a_i is defined to be:

$$a_j = f\left(\sum_i a_i \cdot w_{i,j}\right)$$

For example, if neuron 2 impinges on neuron 4 (neurons 1 and 3 not shown):



then these neurons can implement 'and', 'or' and 'not' connectives in propositional logic:

p:Rainy	q:Cloudy	p and q
False	False	False
False	True	False
True	False	False
True	True	True

p:Rainy	q:Cloudy	p or q
False	False	False
False	True	True
True	False	True
True	True	True

p:Rainy	not p
False	True
True	False

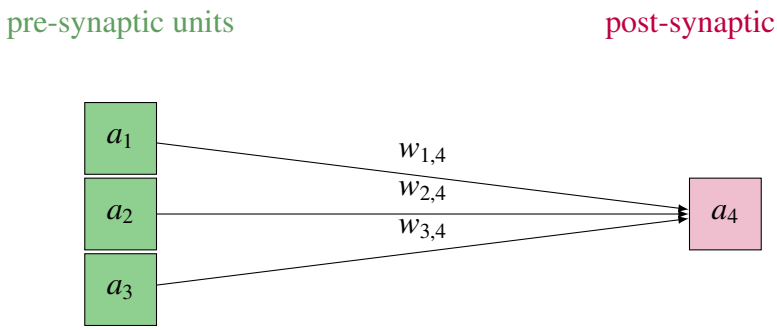
We do this by treating 'False' as 0 and 'True' as 1, then multiplying these by appropriate weights:

- if a_1=1, $w_{1,4}$= − .5, $w_{2,4}$=1, $w_{3,4}$=1 this forms an 'or' neuron (fires if ***any*** antecedent fires),
- if a_1=1, $w_{1,4}$= − 1.5, $w_{2,4}$=1, $w_{3,4}$=1 this forms an 'and' neuron (fires if ***all*** antecedents fire),
- if a_1=1, $w_{1,4}$=.5, $w_{2,4}$= − 1, $w_{3,4}$=0 this forms a 'not' neuron (fires if antecedent a_2 does not).

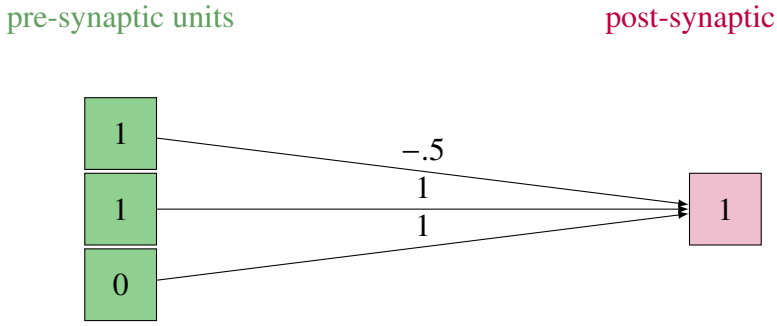
This allows a neural network to implement any propositional logic expression.

1.4 Simplified graphical representation

We can draw a neural network more simply like this:

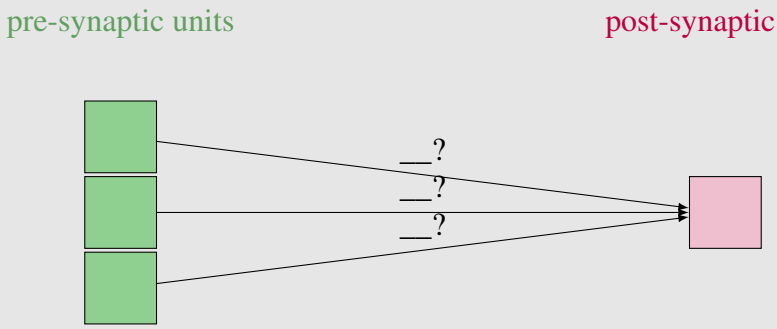


or with values filled in, for the 'or' gate above ($\sum_i a_i \cdot w_i = 1 \cdot -.5 + 1 \cdot 1 + 0 \cdot 1 = .5$, $f(.5) = 1$):



Practice 1.1:

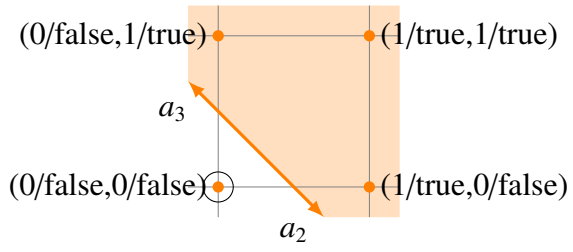
Fill in weights and post-synaptic value for the 'and' gate above, with antecedents a_2=1, a_3=0:

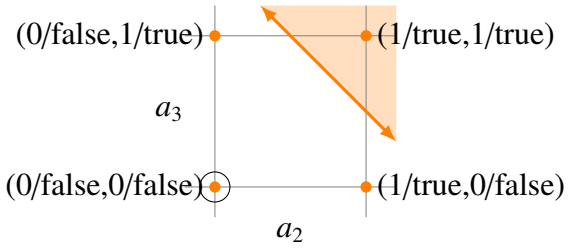


1.5 Multi-layer neural networks

Single-layer networks are just linear separators when graphing input as separate dimensions.

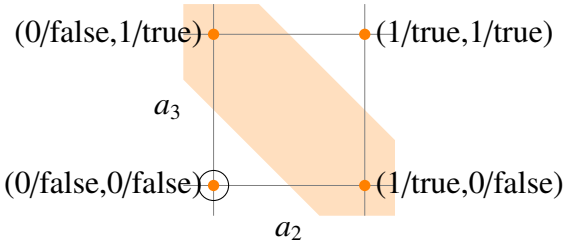
Here are graphs for the 'or' and 'and' neurons from the previous section (orange is output 1):



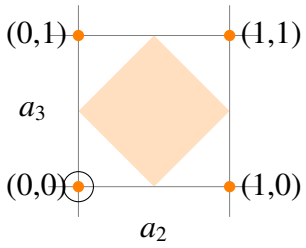


They can't represent:

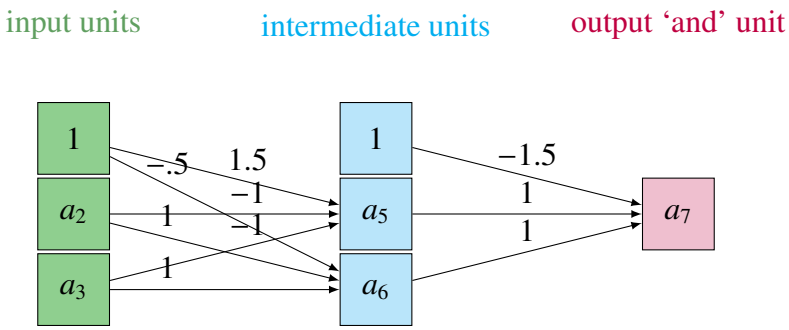
- ‘exclusive or’ (one input is true and the other input is false):



- ‘middle ground’ (excluding peripheral configurations, for neurons with continuous values):



But a multi-layer network can represent these functions (this is ‘exclusive or’):



In fact, two layers can represent *any* propositional logic function in ‘disjunctive normal form’ (that is just a disjunctive list of configurations that yield a ‘True’ output)!

Practice 1.2:

Fill in the values for input a_2=0, a_3=0 in the multi-layer ‘exclusive or’ network above:

References

[Mcculloch & Pitts, 1943] Mcculloch, W. & Pitts, W. (1943). A logical calculus of ideas immanent in nervous activity. *Bulletin of Mathematical Biophysics*, 5, 127–147.