Language Analytics

Session 5 – Text Classification 2 (Neural Networks)

Ross Deans Kristensen-McLachlan

rdkm@cas.au.dk

Course outline

- 1. Introductions
- 2. String Processing with Python
- 3. NLP for linguistic analysis
- 4. Text Classification 1
- **5. Text Classification 2**
- 6. Word embeddings
- 7. Language modelling 1
- 8. Language modelling 2
- 9. BERT
- 10. More BERT
- 11. Project pitches
- 12. Generative models
- 13. Social impact

Plan for today

- Catch up

- 1. What is a neural network?
  - From logistic regression to NNs

- 2. Code-along session
  - Simple NNs using scikit-learn
  - Python scripting

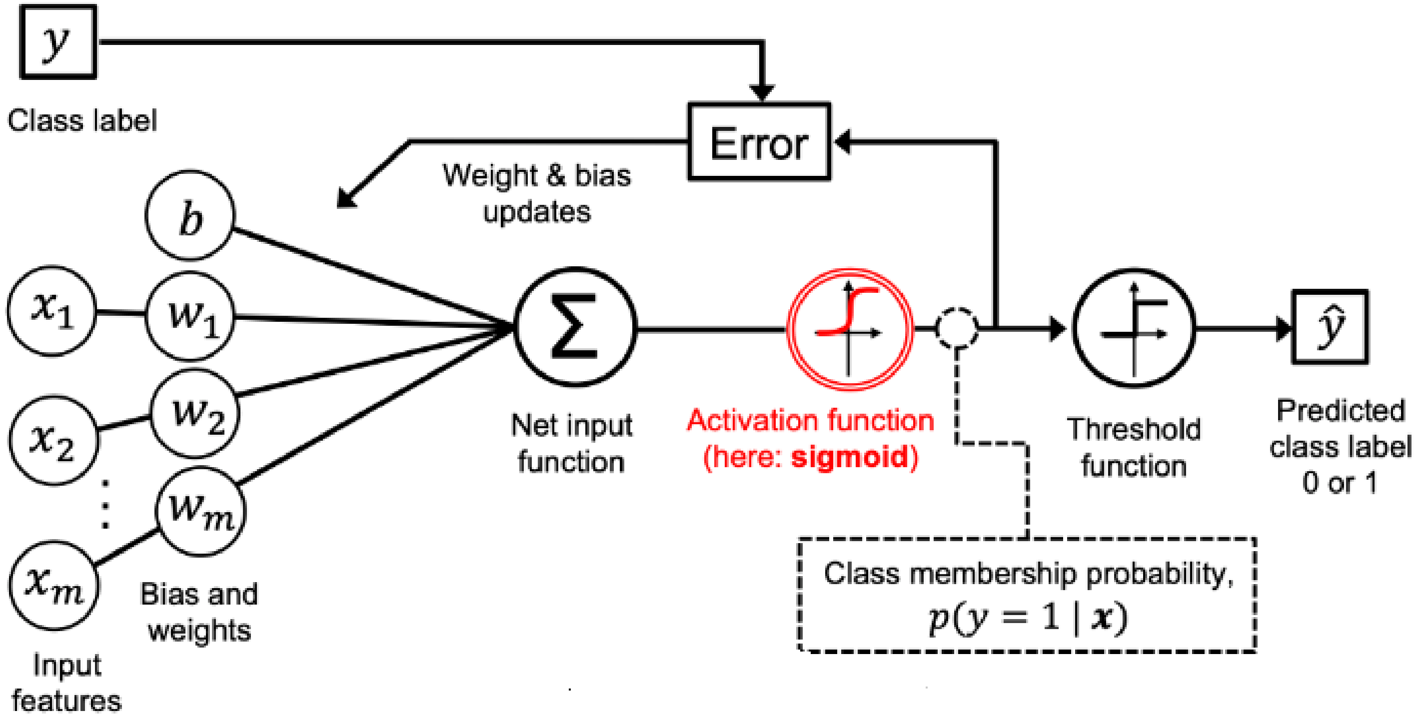
Logistic regression classifier

- A logistic regression classifier does the following:
  - Takes some input features from the training data
  - Estimates parameters to best fit the model
  - Uses these parameters and probabilities to predict class membership in the test, given some decision boundary

- In the case of text classification, what are the input features?

- How does the model learn the parameters?

Logistic regression classifier



How the classifier learns

- A logistic regression classifier is used to learn parameters to estimate the probability of some class or category, given some input features

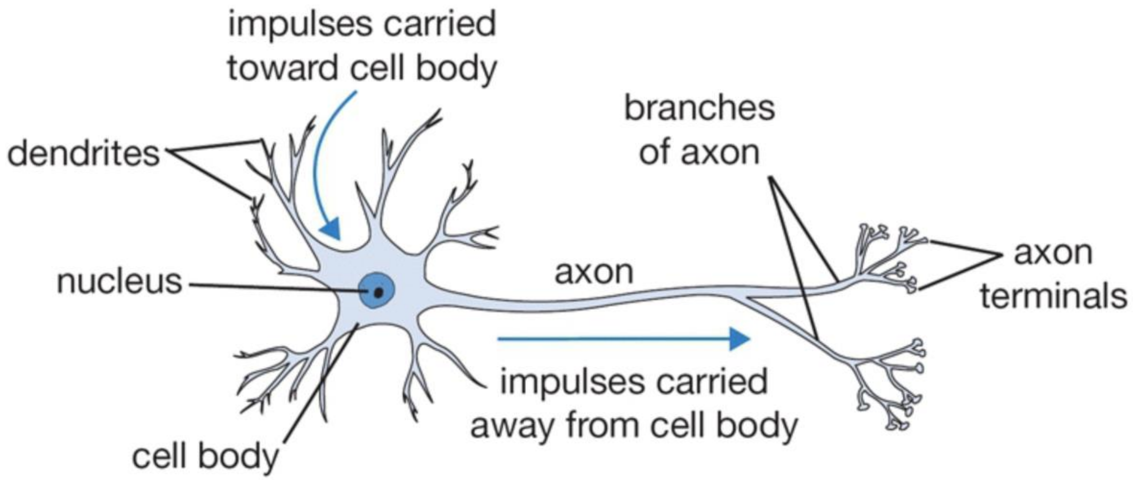
- These parameters are *weights* and *biases* are assigned to each of the features

- The algorithm learns the best weights by finding the ones which return the smallest difference in the predicted and true labels in the training data

- It does this by finding the weights and biases which minimise some mathematical function, a process which is controlled algorithmically

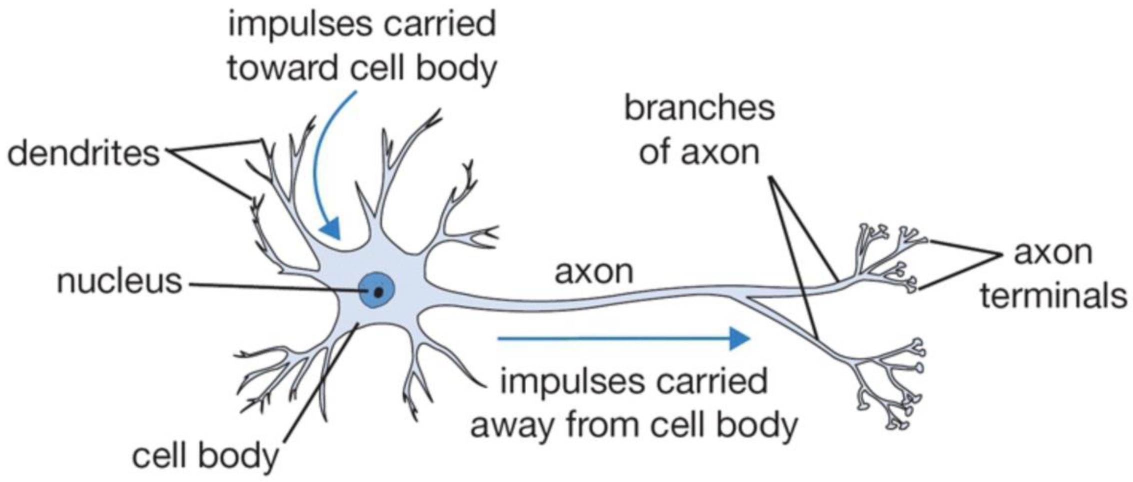
What is a neural network?

- Neural networks are based (very, very loosely) on a biological analogy
- The basic computational unit of the brain is a **neuron**
- ~86 billion neurons can be found in the human nervous system and they are connected with approximately 10^{14} - 10^{15} **synapses**
- Each neuron receives input signals from its **dendrites** and produces output signals along its (single) **axon**



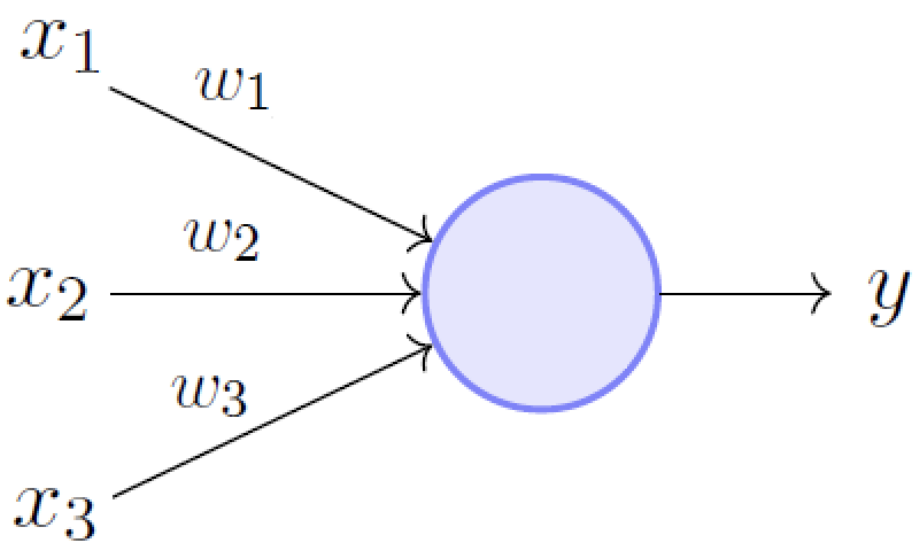
What is a neural network?

- We can think of a neuron as a kind of minimal information processing unit
- It takes some kind of input
- It performs some kind of processing on that input
- It takes the result of that processing and feeds it forward to another neuron or set of neurons
- Using this analogy, we can create a *computational model of a neuron*



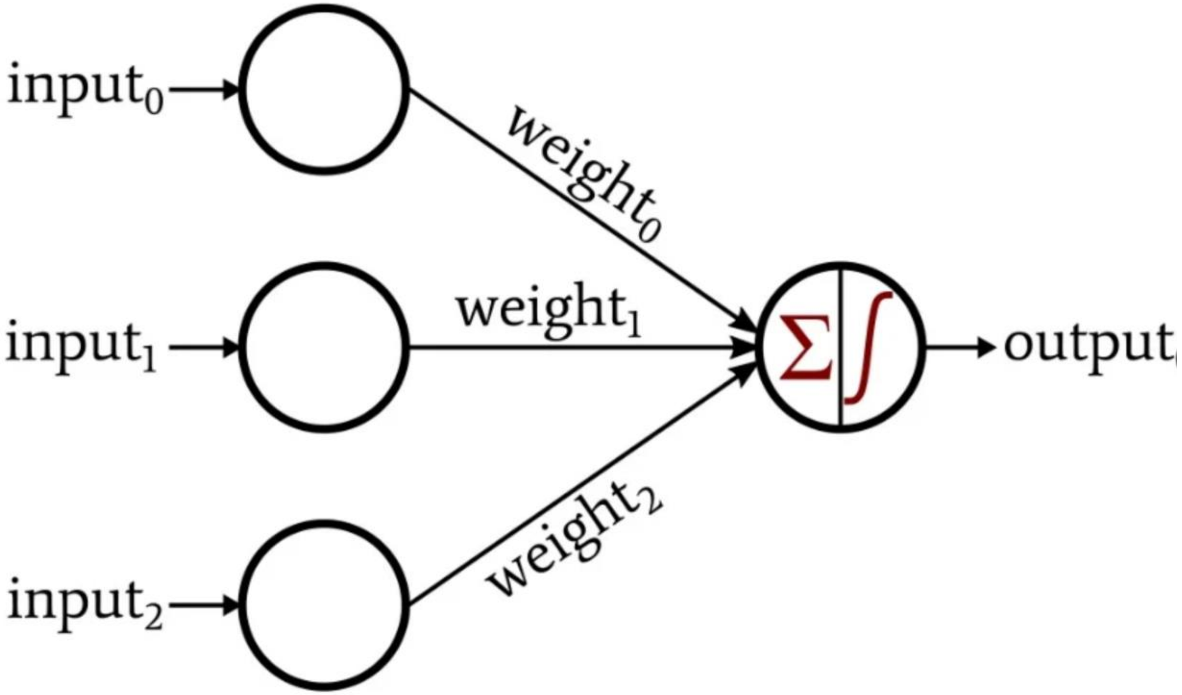
What is a neural network?

- Here, we've abstracted away from the biology and just consider the flow of information
- We have a series of inputs *X*
- Each of these inputs has some kind of weight *W*
- These values are fed into the neuron which processes the values
- In certain contexts, the neuron *fires* and passes some kind of an output as *Y*
- *But what's actually going on in the neuron?*



A simple perceptron

- Generally speaking, the neuron produces a weighted sum of the input values Σ
- If this value is above a certain threshold, the neuron 'fires'
- This threshold is determined using something called an *activation function* $\int$
- A common activation function is a sigmoid function…
- … does this all look familiar?

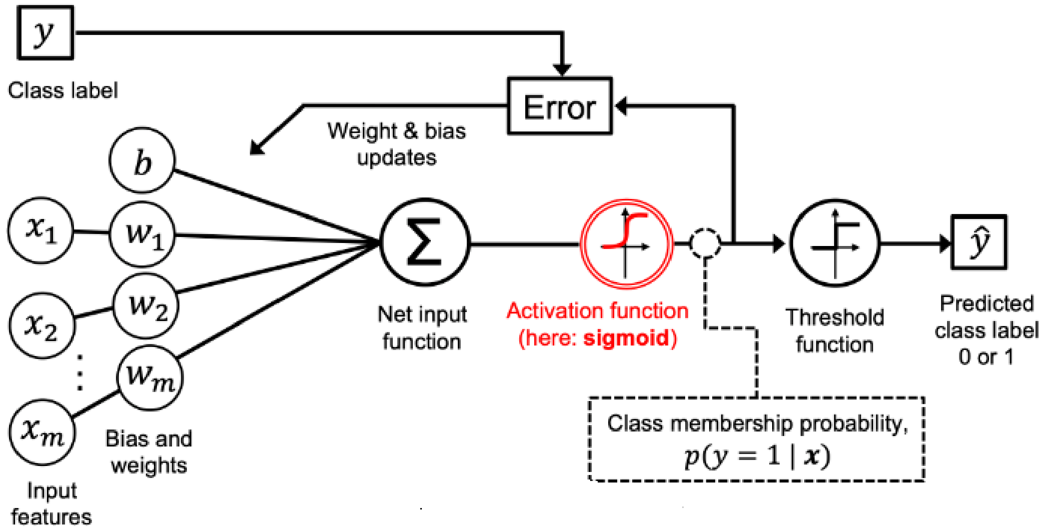


A simple perceptron

- In this example with a single neuron and a sigmoid activation function, what we have is architecturally identical to a logistic regression classifier

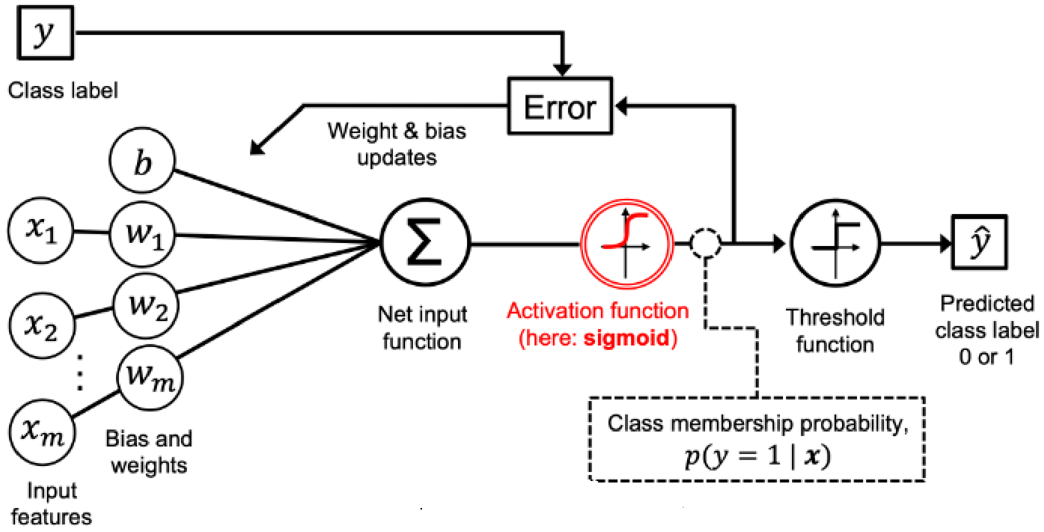
- Minor difference: the unit-step function which forces the output from the activation function to be either 0 |1
    - Think of decision boundaries

- Other activation functions are available and work in slightly different ways – we'll see more in the coming weeks



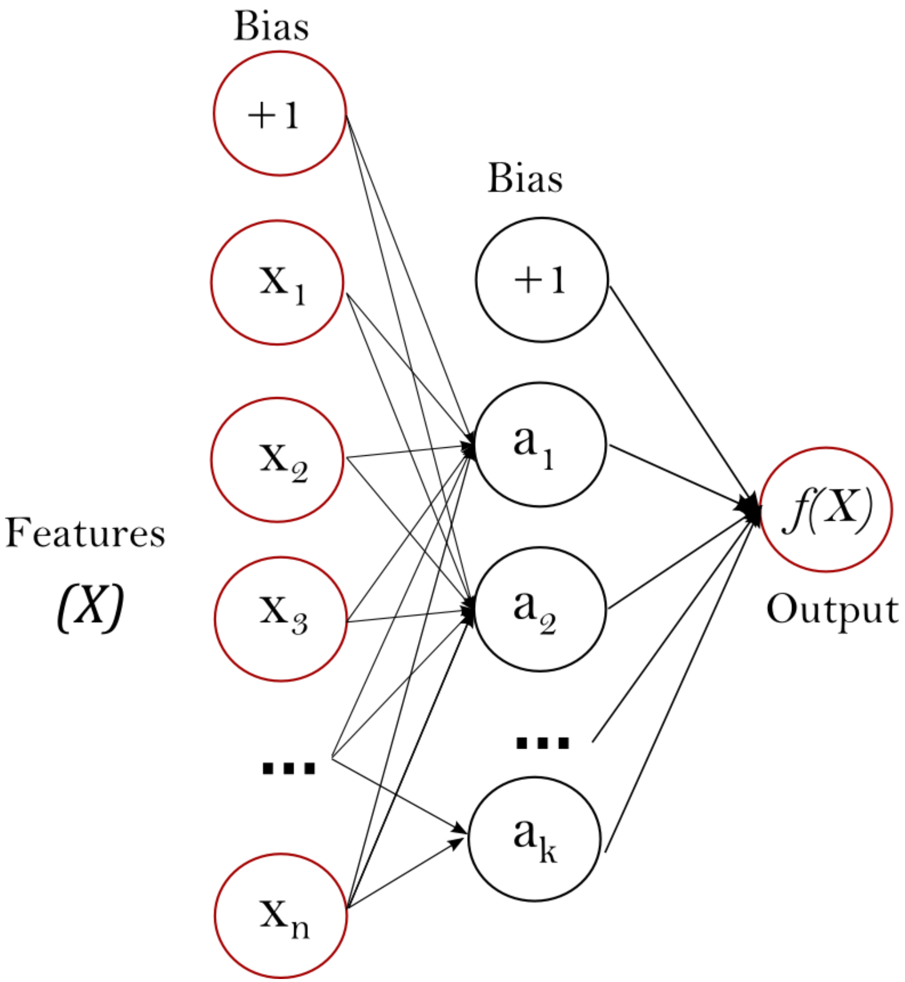
A simple perceptron

- A neural network is just a system of weights and biases learned from data
- In this case, information flows from inputs to outputs in one direction
  - This is a called a *feed-forward network*
- The error from the model is fed back to the start to adjust weights
  - Known as *backpropagation*
- The weighted inputs are fed through the network until optimal results are reached through minimising a loss function



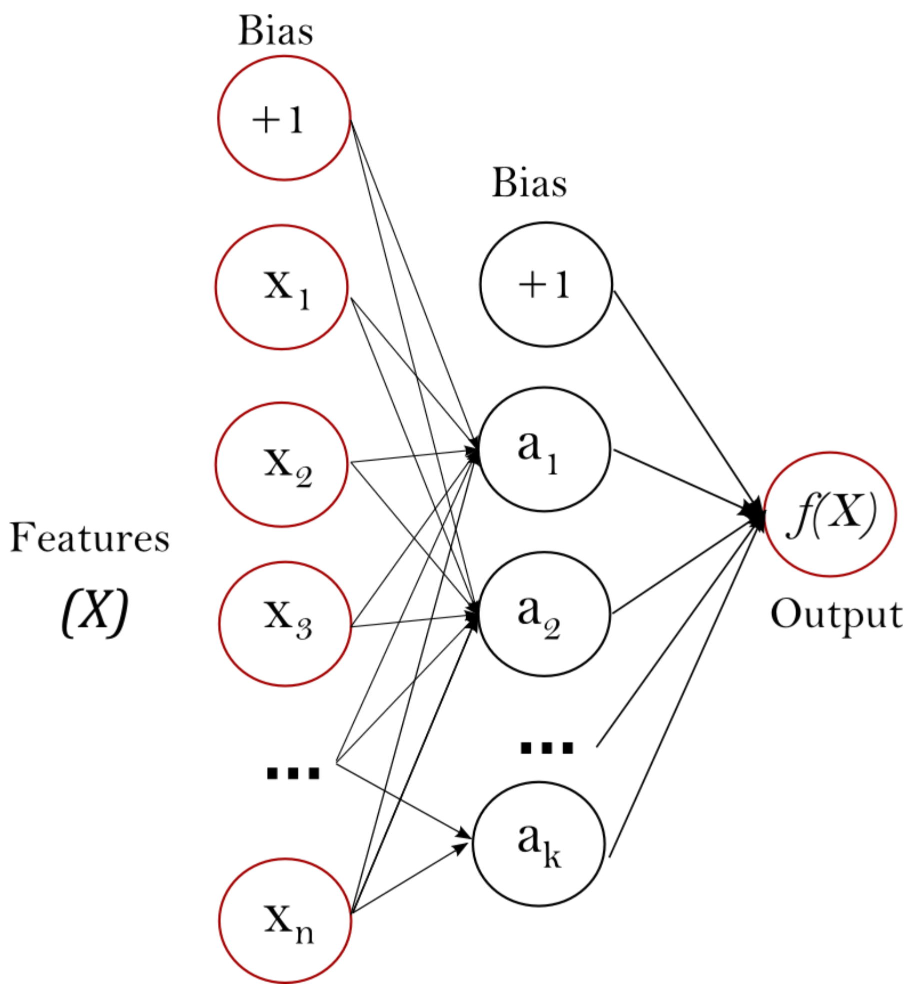
From single neurons to networks

- At the absolute simplest, a neural network is literally what the name suggests – it is a *network* of *neurons*
- Specifically, we introduce a hidden layer or layers of neurons between the input and output layer
- In this neural network, each input corresponds to some feature in the data
  - Value in a BoW vector of Tf-IDF vector
- Each output corresponds to label in the data
  - So this is for binary classification
- What about the hidden layer?



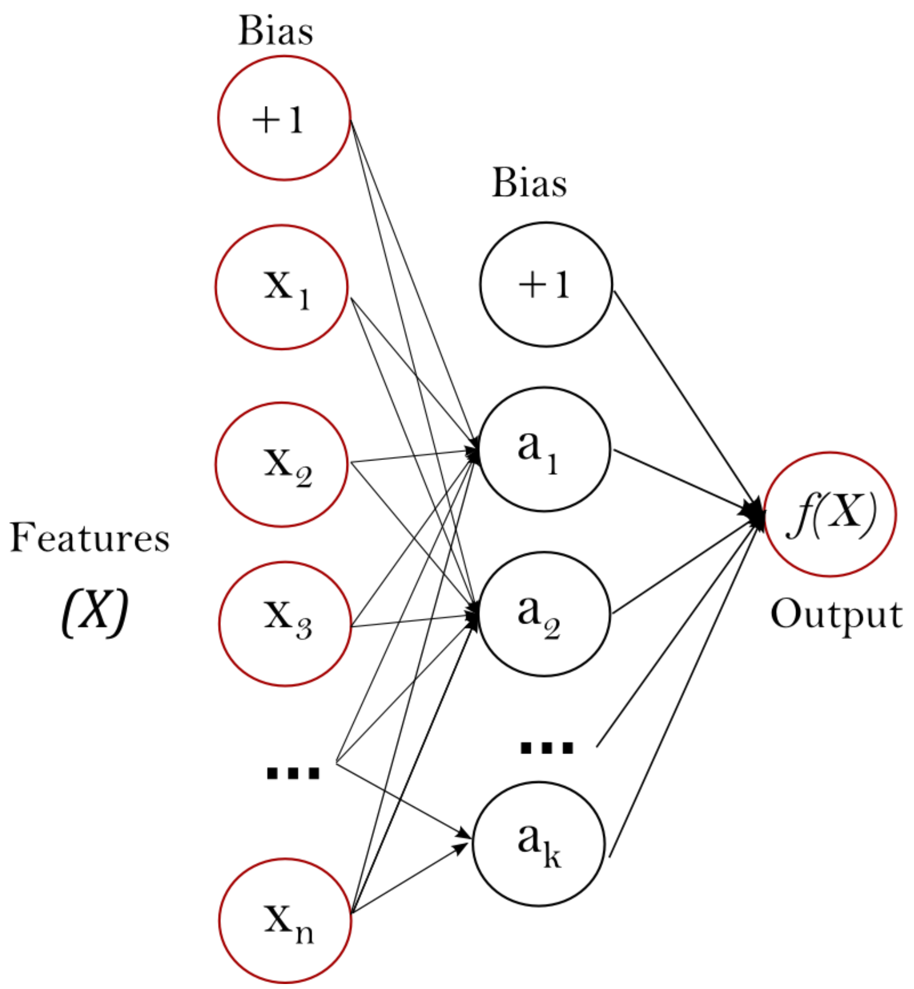
From single neurons to networks

- The hidden layer is simply a number of individual neurons of the kind we've looked at
- Each blue node puts the sum of some weighted inputs and runs them through an activation function, producing an output
- Notice how each input is connected to each node in the hidden layer
- Each node in the hidden layer is connected to both output node
- This is a *fully-connected, feed-forward network*



From single neurons to networks

- The network takes labelled input data and learns a which fits the data based on the labels
- Notice that there is no interaction between nodes of the same layer
- Essentially each node is learning some (linear) aspect of the data independently of the others



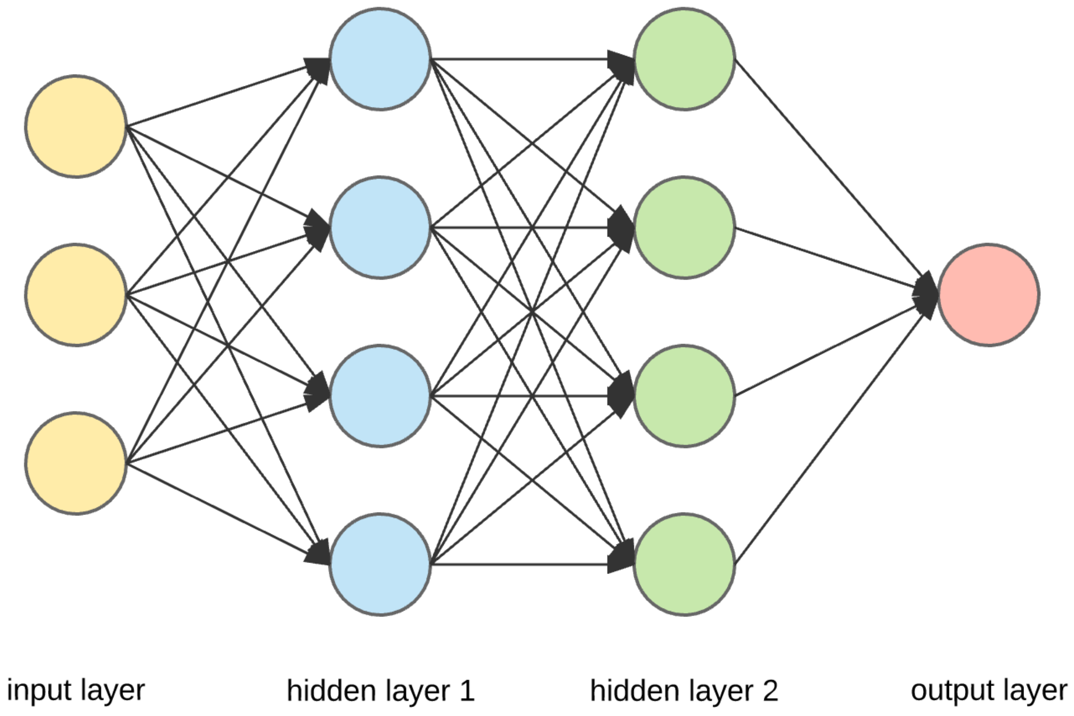
Multilayer networks

- In principle, neural networks can have any number of hidden layers and nodes

- Be aware of how quickly this can grow in terms of computational complexity!

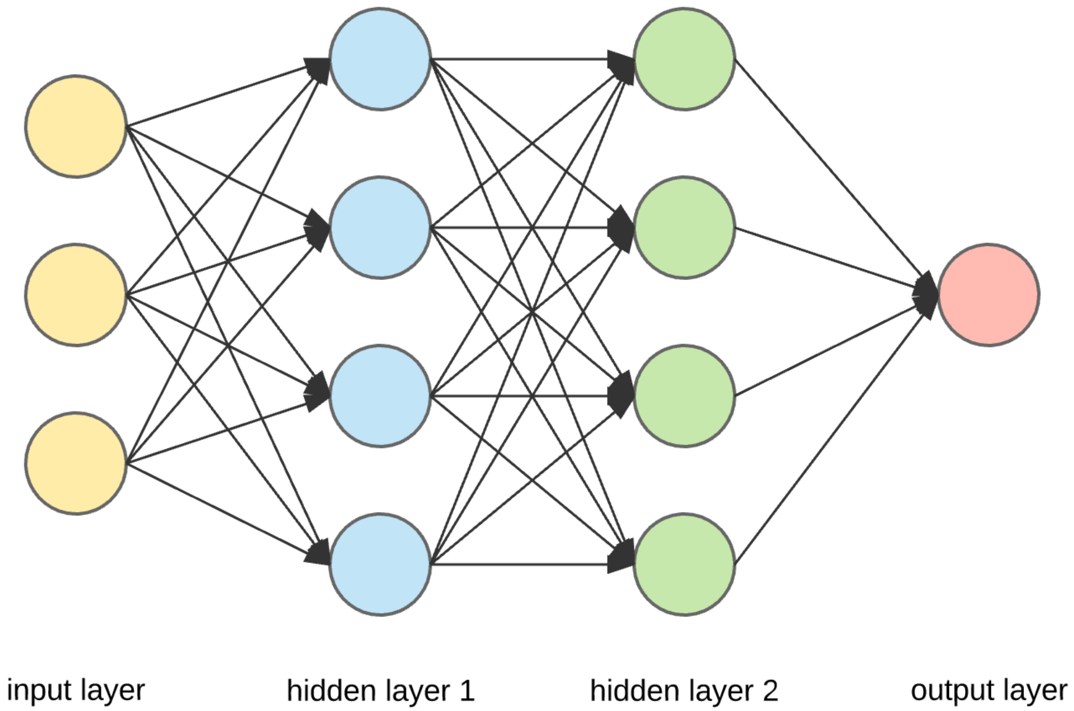
- Imaging an input layer of 1000 nodes
  - Hidden layer 1 = 500 nodes
  - One output node

- Q: How many weights weights would this model have to learn?



Does size matter?

- In general, for this kind of network, we don't really need to go beyond 3 hidden layers
- Mathematically, 3 hidden layers is enough to approximate any arbitrary decision boundary for any given classification problem
- The essential point is that neural networks are *universal function approximators*
- This makes them a powerful computational method for modelling complex, non-linear patterns in data



Summary

- Neural networks draw loosely on an analogy of how neurons function in the brain

- A computational neuron takes some weighted inputs; calculates their sum; puts this through an activation function; and produces an output

- Neurons can learn from their errors through a process known as *backpropagation*

- These neurons can be ground together in hidden layers, forming a neural network

- The simplest kind of architecture is what we're looking at today: *a fully-connected, feed-forward neural network*

Break

And head over to UCloud