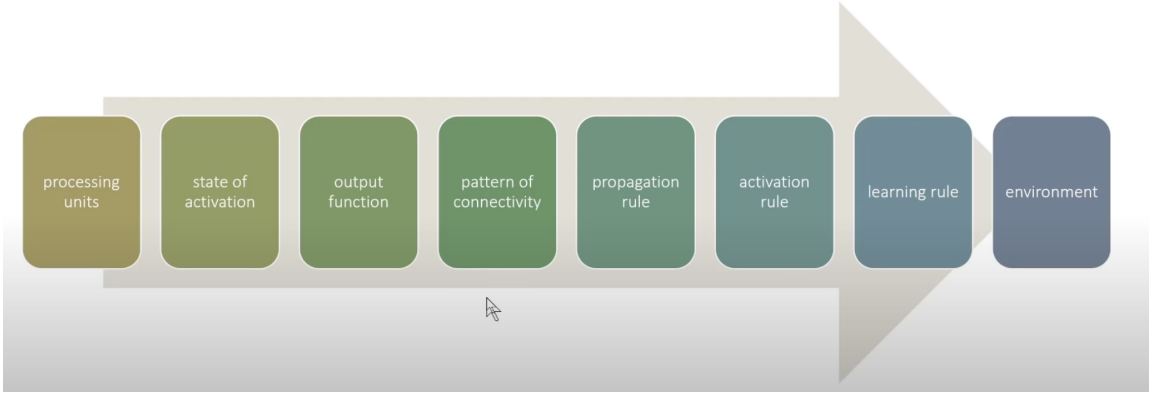
Lesson 1

- Started in 1943, with a mathematical model of a neuron
- 1953: Mark 1 Perceptron
- AI Winter: Minsky and Papert wrote a book that said a single layer of perceptrons couldn't model an XOR gate. Multiple layers could, but this was largely not noticed.
- 1986: Parallel Distributed Processing

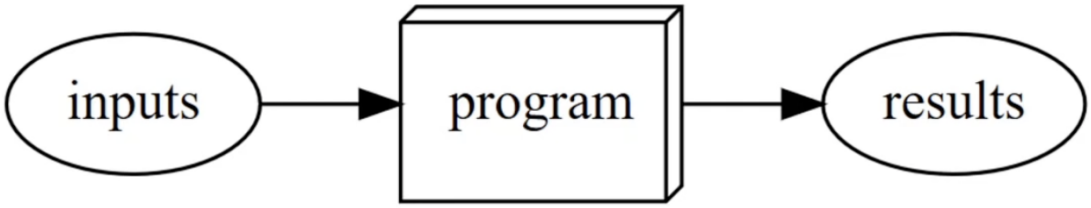


- Adding just one extra layer of neurons is enough to allow any mathematical model to be approximated with a neural network.
  - In practice, slow and big
  - to get practical performance, they require even more layers of neurons
  - Deep instead of wide

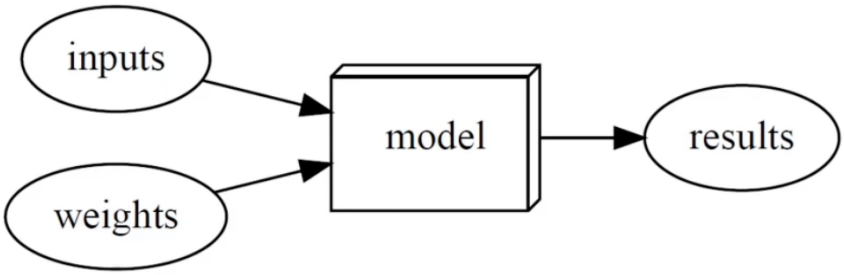
- Software stack: fastai → PyTorch → Python
  - fastai is designed for teaching + development
  - layered API
  - Ability to switch to Keras + TF
- Remember to discuss on: forums.fastai.com
- Randomness involved in training a model

- Don't expect same results every time

- In regular programming

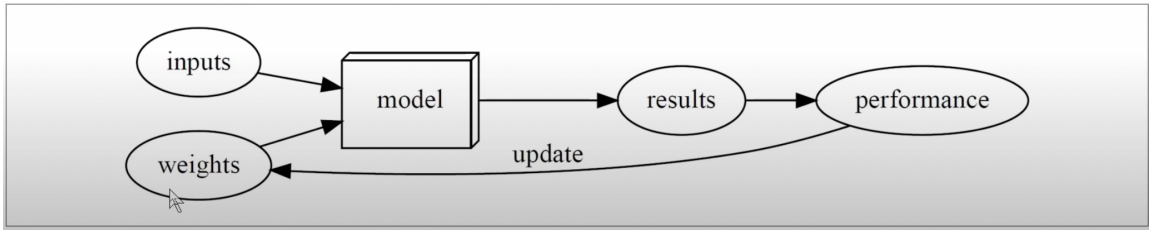


  - doesn't work very well for recognising pictures
- Arthur Samuel pioneered: instead of telling the exact steps required to solve a problem, show examples of the problem to solve and let it figure it out itself

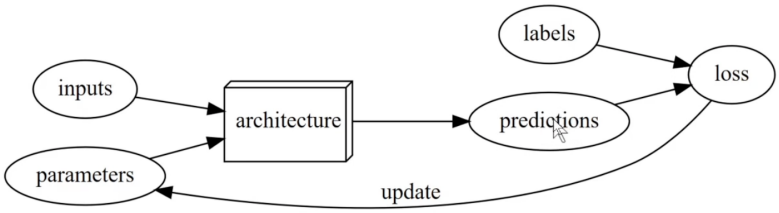


> *Suppose we arrange for some automatic means of testing the effectiveness of any current weight assignment in terms of actual performance and provide a mechanism for altering the weight assignment so as to maximize the performance. We need not go into the details of such a procedure to see that it could be made entirely automatic and to see that a machine so programmed would "learn" from its experience.*

  - Not just depend on inputs, but also a set of weights (parameters to alter the way to solve the problem), then we can alter the weights to find out the best way to solve the solution



- Requires a function in $model$ that can approximate any problem using different set of weights depending on the weights. Neural network is that function. *Universal Approximation Theorem* proves that a neural network can be used to solve any problem to any level of accuracy.
- A general way is required to update the weights. This is Stochastic Gradient Descent (SGD).
- The terminology has been updated



  - functional form of model → architecture
  - weights → parameters
  - predictions are calculated from independent variables (data without labels)
  - results → predictions
  - measure of performance → loss

- Learning approach only creates predictions, not recommended actions
- data needs labels
- Positive feedback loop
  - original data is biased making the model biased
  - more the model is used, more biased the data becomes

    - making the model even more biased
- Error rate is always calculated on validation dataset
    - to prevent overfitting
    - if error rate were calculated on training dataset, it would be optimised for that dataset and not work well on other dataset