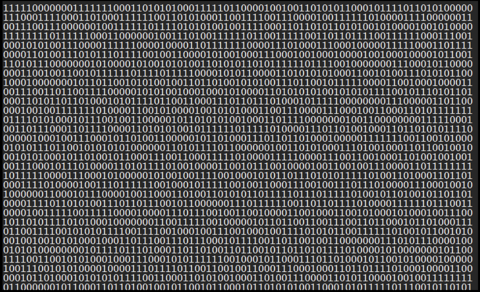
Using Machine Learning to tell if a Spider Died

Jack Bosco

February 14, 2023

Goal

- Want to learn about circadian rhythms in spiders
  - How does a usual (12L:12D) cycle compare to a (12D:12D) cycle?
- We have large sets of data to help us determine this



```
1111100000000111111100011010101000111110110000100100110101011000101111011010100000
1110001111000110100011111100110101000110011110011100001001111110100001111000000011
001110011100000010011111101111010101001001111000110110101101010010100001001010000
1111111101111111000110000001001110100111111011001111100110110111100111111000111001
0001010100111000011111100001000011011111111000011101000111000100000111110001101111
000011010011101011101111001001100001010010001110001001000100001001000100001011001
11010111000000001010000101001010100110101011010111111101111001000000011100010110000
000110010011001011111101111011111000010101100001101010101000110010100111010101100
100010000000101101100101010010011011010010101001110110010111110000110010001000011
0011100110110011110000001010100100010001010000110101010100101010111100101110101101
00011010110110100010101111011001100011101101110100010111111000000000111000001101100
0001001001111111010000110010100001001010100011001110000111000100110001101011111111
011110101000101110010011000000101101010100100011011110000000010011000000000111110001
0011011100011011110000110101010010111111101111101000011101101001000110110101011110
0000010001001110001011010011000001011010001110110110100010000001111111001100101000
010101110110010101010100000011010111101100000010011010100011101001000110110010010
001010100010110100101100011100110001111101000011111000011100110010001101001001001
001110001011101000011010111101001000011001011100100001001100100111000011011111111
1011111000011100010100000010100100111100100010101101110101011111010011010001101101
000111101000010011101111110010001011111001001100011100100111011101000011100010010
100000011000101110000100110001101001101010110111110111011111010010110100101101101
000011110110101001110110111001011000000111011111100110110111101000011111101110011
000010011111001111100001000011101110010011001000011001000110010100010100010011100
101101011110101000100000000110011111100100000010110110011001110011011000101101000111
011001111001010101111001111001000100111001000100111101010110011111101001011001010
0010010010101000100011011100111011100010111100110110010011000000001110101110000100
0101010000000001011110111010001101101001101100101101101011110100001010000000101100
1111001100101010001000111000101011111110010001011000111011010001011001010000100000
100111001010100001000011101111011001100100110001110001000110110111101000100001100
000101101000101010101111001100011010100100011010011100001101011000010010011111111
011000000101100011011010010010110010110001011010101001100010101111101110010110101
```

Problem

Null Character

Inaction (spider not doing anything) is coded similarly to sleeping.
Actually, its exactly the same

- As is, no human can look at this data and determine if a spider was affected by change in day/night cycle or if it died overnight
- This biases the data, rendering tons of information inconclusive

Problem Redefined

Dead|Not-Dead

`A classification algorithm` *`should`* `be able to use pattern matching and see when or if a spider died.`

For an input string of n 0s or 1s...

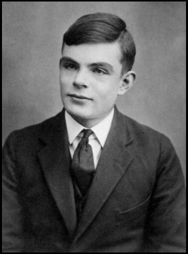
- Action Space: n. We should be able to point (somewhat accurately) to the moment when the spider died.
- State Space: 2^n. Every possible arrangement of 0s and 1s needs to be classifiable. This number is HUGE.

Our state and action space makes training **Dead|Not-Dead** non-trivial.

Can we do it?

Yes

If the problem is solvable, then a perceptron can learn to solve it



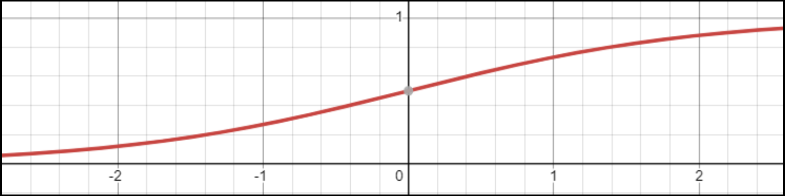
Not So Fast...

Just because it can doesn't mean it won't take forever

So not only do we need to train an algorithm that solves this problem, we need to be smart about it as to not end up spending $1,345,232,923 on AWS.

Solution 1: ‘Meh’ Classifier

Multilayered Perception (MLP)



- a sigmoidal (non-linear) activation function
- outputs on a gradient
  - 0.0 → The spider died (earlier)
  - 1.0 → The the spider is totally still alive
  - $0.25 < x < 0.75$ → ‘Meh’

Solution 1: ‘Meh’ Classifier

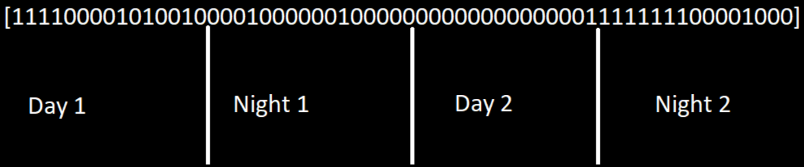
Pros

- Simple, understandable, we’ve had these since 1989
- I can (and have) implemented this with only numpy
- quick/inexpensive

Cons

- results may be uninformative
- Vanishing Gradient Problem - we can’t to better than ‘meh’

Pre-Classification



[111100001010010000100000010000000000000000001111111100001000]

Day 1

Night 1

Day 2

Night 2

- split the dataset into *m* groups
- implement *m* hidden layers for the perception and do **much** better than ‘meh’
- Unfortunately, with sigmoidals this is not possible (Vanishing Gradient)

Solution 2: We Go Deeper

Rectified Linear Activation Function (ReLU)

A piecewise function with constant derivative (linear). It's just a line with a cutoff!

Overhaul sigmoidals and use a ReLU instead

- Solve vanishing gradient problem → more hidden layers!
- unlock more sophisticated 'deep' learning algorithms (CNN, A2C)

Solution 2: We Go Deeper

Pros	Cons
▪ more informative and reliable output ▪ I can still code this myself (with numpy)	▪ more hidden layers are more expensive to train

PyTorch

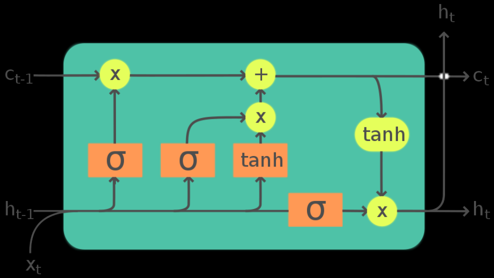
PyTorch Package for Python

"An open source machine learning framework that accelerates the path from research prototyping to production deployment"

PyTorch makes it easy to build things I don't understand!

- A2C - Actor Critic
  - value-based actor (Q-learning)
  - policy-based critic (perceptron)
- CNN - Convolution Neural Net
  - multilayered sparsely connected perceptron
  - GoTo choice for image classification
- LSTM - Long-Short Term Memory
  - recursive neural net with variable-time weight updates

Solution 3: LSTM



- Recursive (not feedforward) Neural Net
  - feedback connections allow for **processing entire sequences of data** - Wikipedia
- Sigmoidal - but somehow still works
  - solves Vanishing Gradient problem by allowing gradients to flow *unchanged* - Wikipedia

Solution 3: LSTM

Pros

- invented in 1995
- Bill Gates likes it
- OpenAI and Deepmind use it
- made for problems like ours

Cons

- might be expensive
- going to require more research
- I definitely couldn't make this without PyTorch