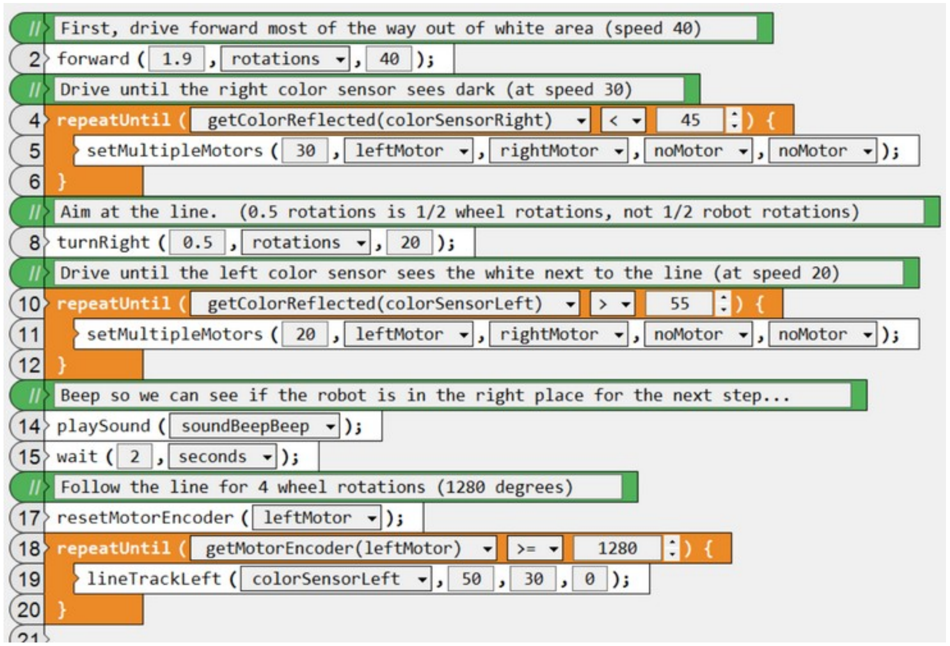
Machine Learning

Techie Pizza #44267
Project, Lesson 4
Michael Lyle

Conventional programming

Normal computer programs are like what we wrote for the robot: a set of instructions to run in order.

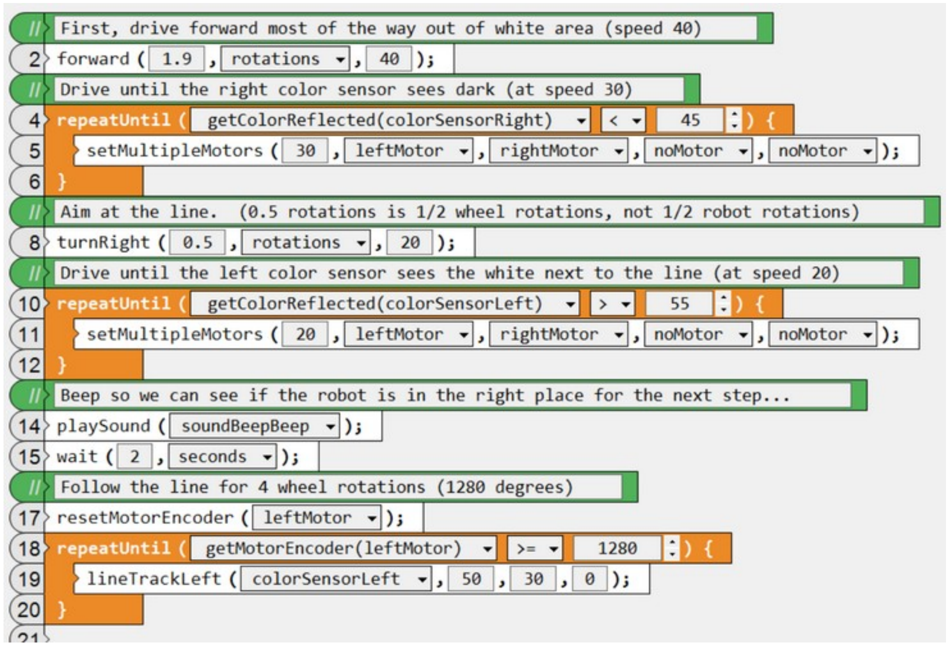


```
// First, drive forward most of the way out of white area (speed 40)
forward ( 1.9 , rotations , 40 );
// Drive until the right color sensor sees dark (at speed 30)
repeatUntil ( getColorReflected(colorSensorRight) < 45 ) {
    setMultipleMotors ( 30 , leftMotor , rightMotor , noMotor , noMotor );
}
// Aim at the line.  (0.5 rotations is 1/2 wheel rotations, not 1/2 robot rotations)
turnRight ( 0.5 , rotations , 20 );
// Drive until the left color sensor sees the white next to the line (at speed 20)
repeatUntil ( getColorReflected(colorSensorLeft) > 55 ) {
    setMultipleMotors ( 20 , leftMotor , rightMotor , noMotor , noMotor );
}
// Beep so we can see if the robot is in the right place for the next step...
playSound ( soundBeepBeep );
wait ( 2 , seconds );
// Follow the line for 4 wheel rotations (1280 degrees)
resetMotorEncoder ( leftMotor );
repeatUntil ( getMotorEncoder(leftMotor) >= 1280 ) {
    lineTrackLeft ( colorSensorLeft , 50 , 30 , 0 );
}
```

Conventional programming

They may have loops; they may store and operate on data; ***but:***

They require a human to anticipate ***exactly*** how to perform the task.



```
// First, drive forward most of the way out of white area (speed 40)
forward ( 1.9 , rotations , 40 );
// Drive until the right color sensor sees dark (at speed 30)
repeatUntil ( getColorReflected(colorSensorRight) < 45 ) {
    setMultipleMotors ( 30 , leftMotor , rightMotor , noMotor , noMotor );
}
// Aim at the line.  (0.5 rotations is 1/2 wheel rotations, not 1/2 robot rotations)
turnRight ( 0.5 , rotations , 20 );
// Drive until the left color sensor sees the white next to the line (at speed 20)
repeatUntil ( getColorReflected(colorSensorLeft) > 55 ) {
    setMultipleMotors ( 20 , leftMotor , rightMotor , noMotor , noMotor );
}
// Beep so we can see if the robot is in the right place for the next step...
playSound ( soundBeepBeep );
wait ( 2 , seconds );
// Follow the line for 4 wheel rotations (1280 degrees)
resetMotorEncoder ( leftMotor );
repeatUntil ( getMotorEncoder(leftMotor) >= 1280 ) {
    lineTrackLeft ( colorSensorLeft , 50 , 30 , 0 );
}
```

In normal programming, we need to be able to break the task up into simple steps.

But some things are hard to break up into simple steps:

“Is there a kitten in this picture?”

In normal programming, we need to be able to break the task up into simple steps.

But some things are hard to break up into simple steps:

“Is there a kitten in this picture?”

Algorithms and heuristics

An algorithm is a step-by-step set of instructions for the computer to perform in order to correctly complete a task.

An example is sorting a list of names by swapping names when they are in the wrong order.

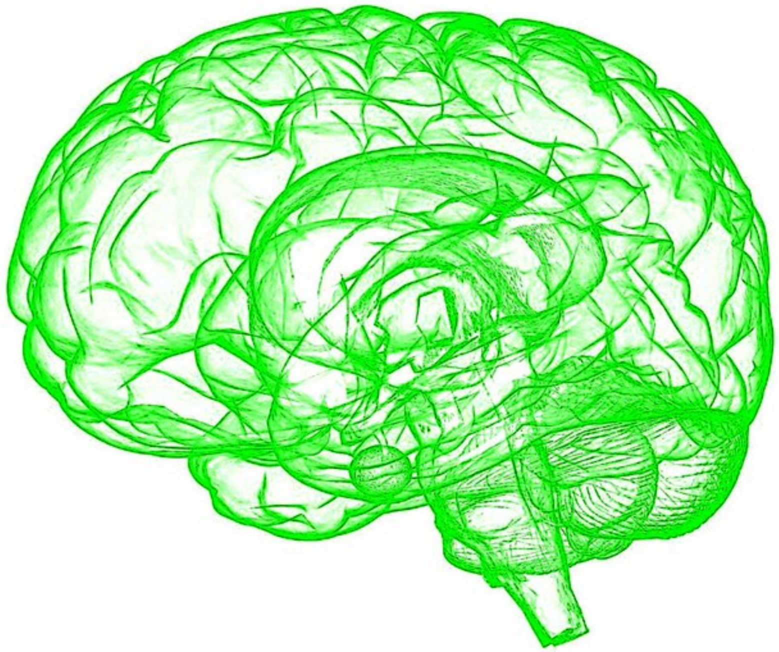
A heuristic is a "rule of thumb" in a conventional program that often gets an acceptable answer.

For instance, if we are trying to find a path somewhere, we can assume we need to move towards it.

We could come up with a set of algorithms and heuristics to tell the difference between sidewalks and streets, but it would probably require advanced math and lots of work.

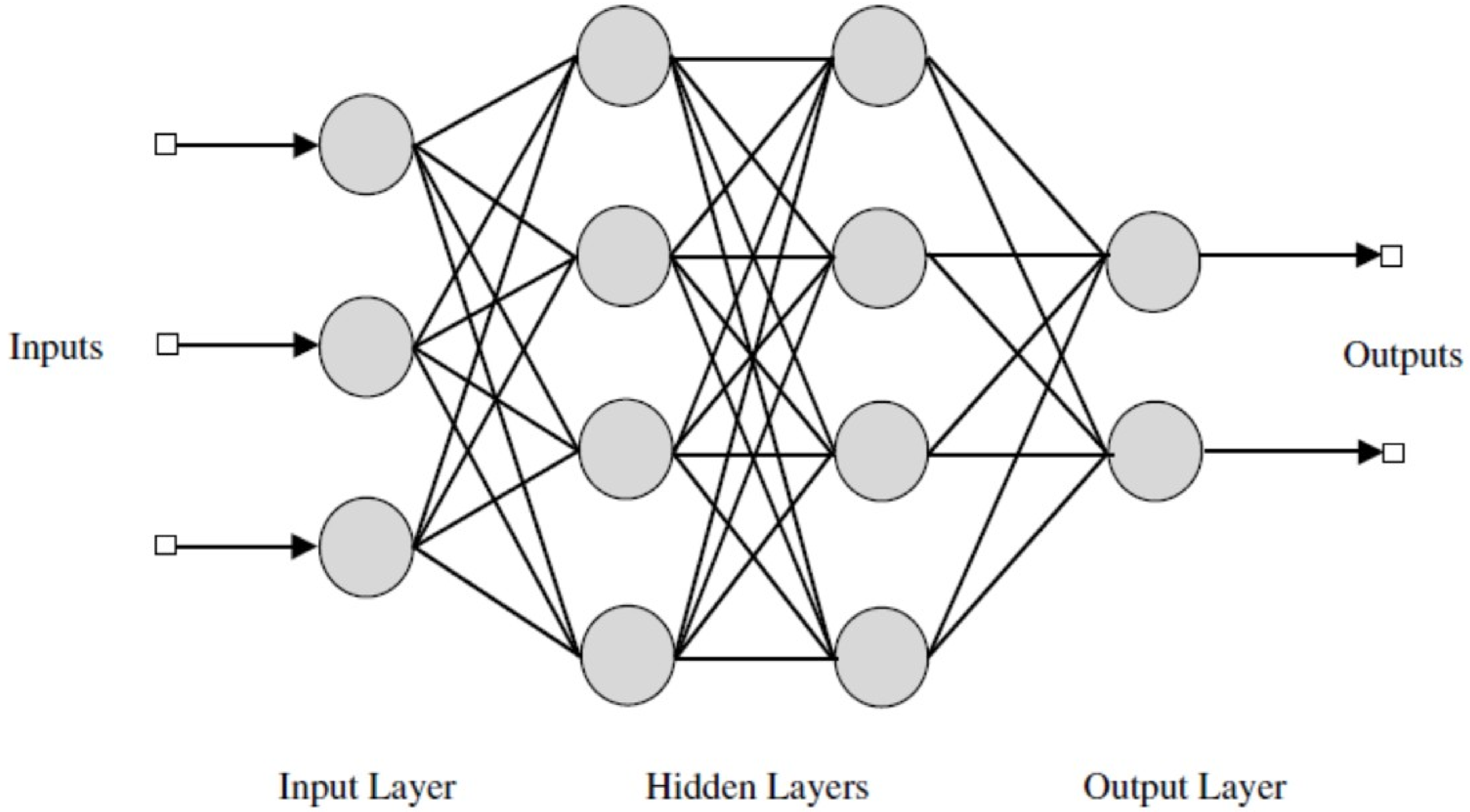
Not how humans do it

- We almost never follow explicit algorithms and simple heuristics.
- We’re willing to override rules / change procedures when they cause bad results.
- We can easily teach people to do tasks that are very hard to write programs to do...



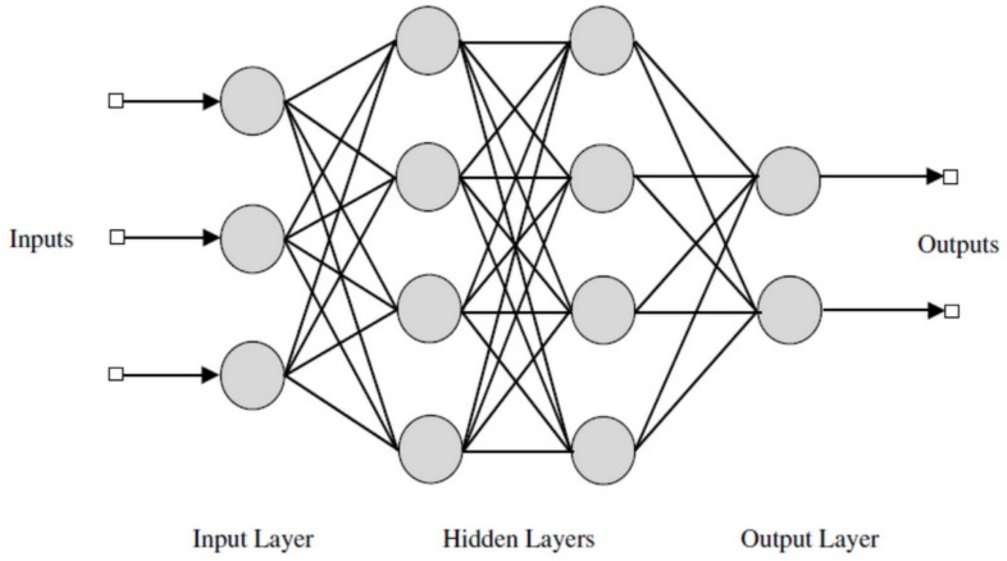
What if we could just teach the computer?

There are many types of machine learning, but we’ll talk about neural networks.


Inputs
Outputs
Input Layer
Hidden Layers
Output Layer

Training an artificial neural network

1) Start with example data and a set of "correct answers."
2) Adjust how strong the connections are to make the neural network produce closer to the output we want. ("Training")
3) Repeat. A lot.
4) For some problems, we may get a result that's as good as a human, or even better!

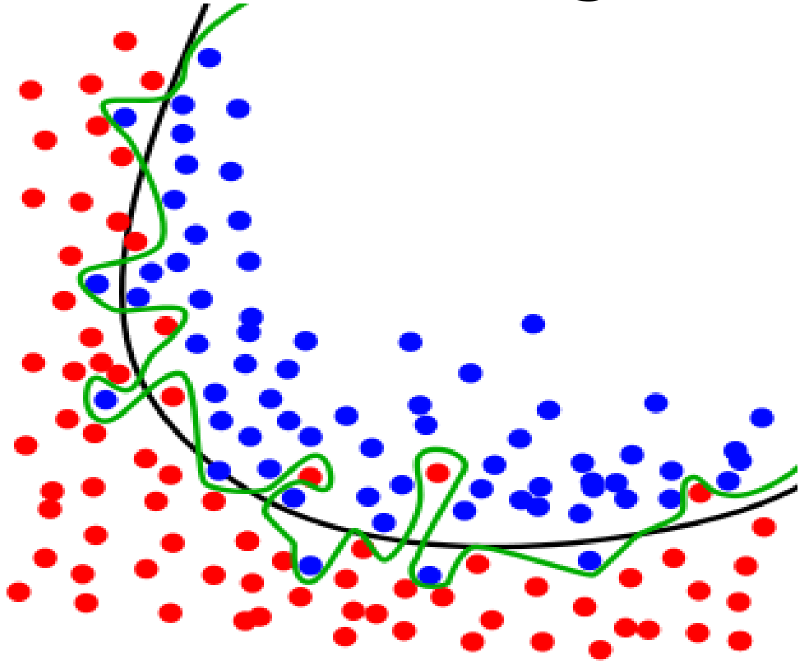


THIS IS YOUR MACHINE LEARNING SYSTEM?
YUP! YOU POUR THE DATA INTO THIS BIG PILE OF LINEAR ALGEBRA, THEN COLLECT THE ANSWERS ON THE OTHER SIDE.
WHAT IF THE ANSWERS ARE WRONG?
JUST STIR THE PILE UNTIL THEY START LOOKING RIGHT.
DATA
ANSWERS

But, be careful…

We may train the system to determine the wrong thing (“Russian tank?” story).

We may train the system in a way that only works well on our training data (“overfit”).



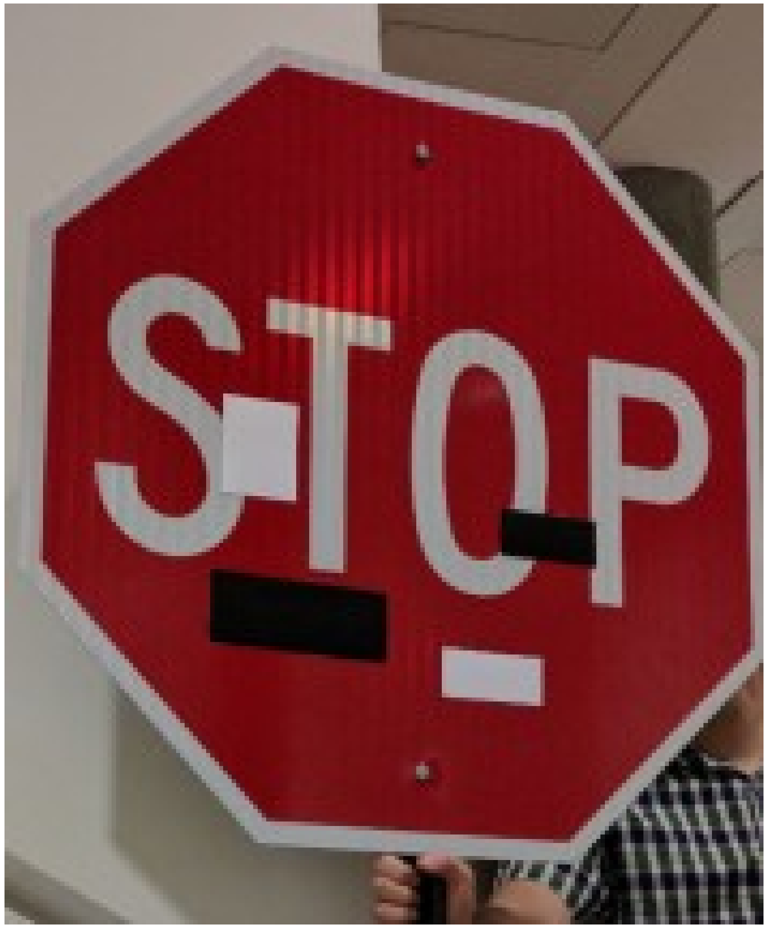
Data, data, data

We need lots of high quality, varied examples to avoid problems in training.

To help fight overfit, we need to use half of our data for training, and then see how well things work on the other half.

Even this practice is statistically misleading, and really only works well for one test cycle. There’s other fancier approaches (cross-validation, regularization, ensembling, other hyperparameters)…

It still doesn’t think like us...



One computer vision program trained to identify street signs in pictures does better than humans on many measures.

But, a sign altered to fool that program convinces it that this sign says
“SPEED LIMIT 55”.

It **really** doesn’t think like us…

- Researchers at Google did an experiment called “Deep Dream” with a neural network trained to recognize things in images.
- It’s kind of the opposite of training: start with a picture, and then alter the picture over and over so it is more like what a neural network expects for a given answer.
- What if we start with a picture of spaghetti, and alter it so that the computer program is more and more certain that it contains a dog?

Despite all the problems...

Machine learning is the best tool we have to solve problems that are too hard to solve with conventional programming.