BASICS OF MACHINE LEARNING

Quick note:

This is *not* a talk on how to build a machine learning program to do task X.

If you want to do this, there are a million tutorials on the internet.

TODAY, I'M GOING TO TRY TO GIVE YOU A DEEPER UNDERSTANDING OF MACHINE LEARNING: WHY IT EVEN WORKS, AND HOW IT RELATES TO OTHER PROGRAMMING WE'VE DONE.

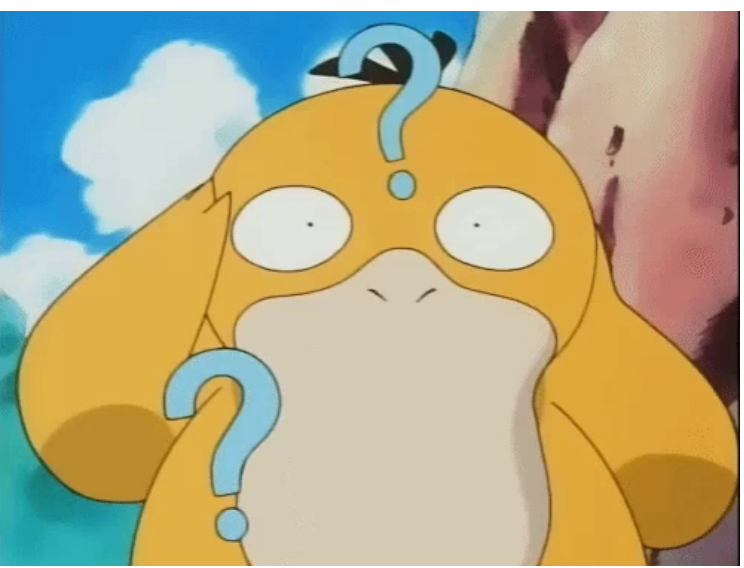
I'll be teaching today with the ultimate goal of getting to a deep neural network, but the principles are very similar for a large number of machine learning tools.

WHAT IS MACHINE LEARNING?

WHAT IS MACHINE LEARNING?

> ***Machine learning*** *(**ML**) is a field of inquiry devoted to understanding and building methods that 'learn', that is, methods that leverage data to improve performance on some set of tasks*

Wikipedia



CLASSIC PROGRAMMING

Completely specify what the computer should do.

Write code that tells the computer what operations to carry out.

```
1 def my_function(x):
2     return 2 * x + 1
```

MACHINE LEARNING

Partially specify what the computer should do.

Write code that tells the computer mostly what it should do, but leave a few values ambiguous.

```
1  def my_function(x):
2      return a * x + b
```

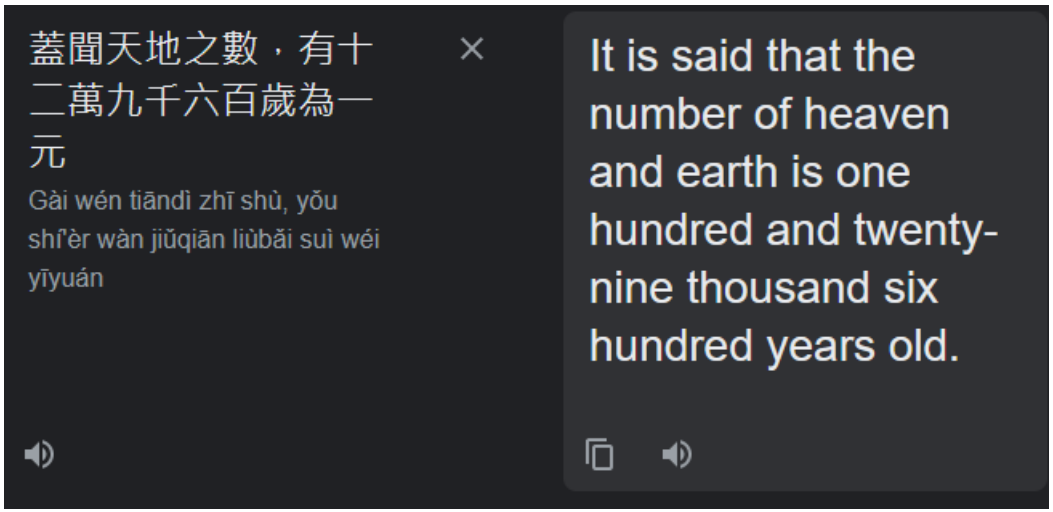
Use examples to **teach** the computer what the **parameters** should be.

MACHINE LEARNING IS NOT SOME MAGICAL WAY TO MAKE COMPUTERS DO THINGS THEY COULDN'T BEFORE!

The computer is still executing a function! Instead of specifying what the function is 100%, we're going to feed the computer examples and adjust the function based on those examples!

FUNCTIONS????????

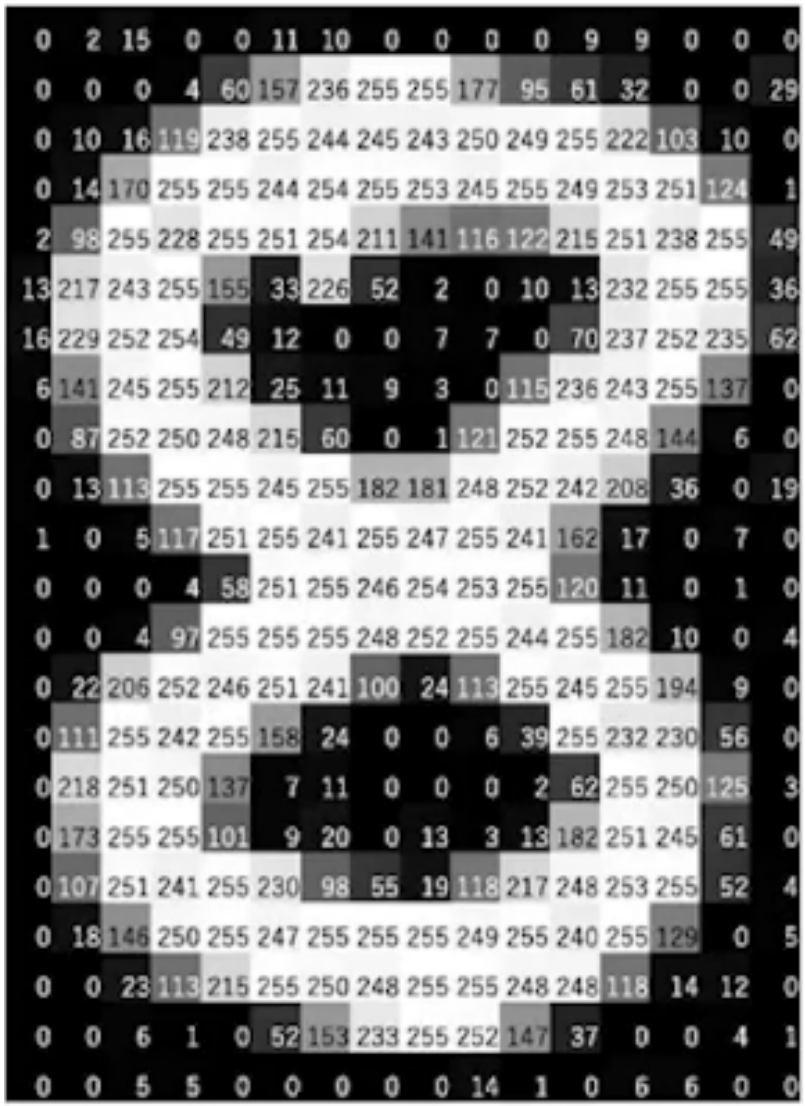
COMMON MACHINE LEARNING TASKS



These are functions???

EXAMPLE: SEMANTIC SEGMENTATION

What type of data is a grayscale (black & white) image?



What would a function look like to determine if a given pixel is part of the stroke or part of the background?

```
1 def is_background(image, i, j):
2   if image[i][j] > 5:
3     return True
4   else:
5     return False
```


Image Credit: https://www.analyticsvidhya.com/blog/2021/03/grayscale-and-rgb-format-for-storing-images/

EXAMPLE: SEMANTIC SEGMENTATION

```
1 def semantic_category(image, i, j):
2     # Do some analysis work
3     #
4     #
5     if is_car(analysis_result):
6       return 1
7     elif is_bus(analysis_result):
8       return 2
9     # etc. etc.
```

```
1 def semantic_segments(image):
2     for i in range(len(image)):
        for j in range(len(image)):
          semantic_category(image, i, j)
```

HOW DO WE ADJUST THE PARAMETERS?

```
1  def my_function(x):
2      return a * x + b
```

THESE PARAMETERS DON'T SET THEMSELVES!

First, we need a list of examples:

- 2, 5
- 7, 15
- -3, -5
- 0, 1
- 100, 201

```
1  def my_function(x):
2      return a * x + b
```

Let's start by guessing values for a, b. Maybe we'll say they start at a = 0 and b = 5.

Example Input	**Example Output**	**Our Output**
2	5	5
5	11	5
100	201	5
-3	-5	5

Well....that's not very good.

```
1  def my_function(x):
2      return a * x + b
```

What if we tried a = 0, b = 100?

Example Input	Example Output	Our Output
2	5	100
5	11	100
100	201	100
-3	-5	100

Well....that's not very good either.

Example Input	Example Output	Our Output
2	5	5
5	11	5
100	201	5
-3	-5	5

WHICH IS WORSE?

Example Input	Example Output	Our Output
2	5	100
5	11	100
100	201	100
-3	-5	100

QUANTIFYING ERROR

We need some way to assign a number to the intuitive idea of "how bad the error is."

$$mse(x_t, x_f) = \sum_{i=1}^{N} (x_t - x_f)^2$$

```
1 def mse(true_out, our_out):
2   loss = 0
3   for i in range(len(true_out)):
4     loss += (true_out - our_out) ** 2
```

These are known as *loss functions*. If you go into ML, you'll hear about a lot of these things:

- MSE Loss
- MAE Loss
- Cross-Entropy Loss
- etc. etc. etc.

THE STORY SO FAR

```
1  def my_function(x):
2      return a * x + b
```

PARTIALLY COMPLETE FUNCTION

LIST OF EXAMPLES

Example Input	Example Output	Our Output
2	5	5
5	11	5
100	201	5
-3	-5	5

LOSS FUNCTION

$$mse(x_t, x_f) = \sum_{i=1}^{N} (x_t - x_f)^2$$

THE DESCENT

```
1  def my_function(x):
2      return a * x + b
```

We tried a = 0, b = 5. These were our results.

Example Input	Example Output	Our Output
2	5	5
5	11	5
100	201	5
-3	-5	5

MSE Loss = 38552

If we increased a, would the results be better or worse?

```
1  def my_function(x):
2      return a * x + b
```

a = 0.1, b = 5

Example Input	Example Output	Our Output
2	5	5.2
5	11	5.5
100	201	55
-3	-5	4.7

MSE Loss = 21440.38

LOSS IS 30% LOWER!

We could just keep trying this!

- Try changing a or b a small amount, see how the loss changes.
- If the loss decreases, keep the change.
- If the loss increases, revert the change.
- Keep going until you can't decrease the loss anymore.

But real models don't have 2 parameters, they have 3 million.

To actually try this for 3 million parameters (and then actually have to change things) is too slow.

Instead, rely on another observation: for our given examples, we can compute the loss exactly based off of a and b.

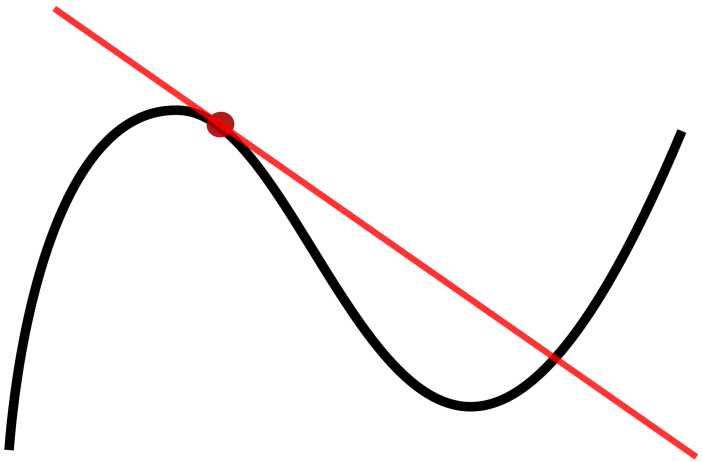
```
1 def compute_loss(inputs, outputs, a, b):
2   loss = 0
3   for i in range(len(inputs)):
4     our_output = a * inputs[i] + b
5     true_output = outputs[i]
6     loss += (our_output - true_output) ** 2
7   return loss
```

For given examples, the loss is a function of a and b!

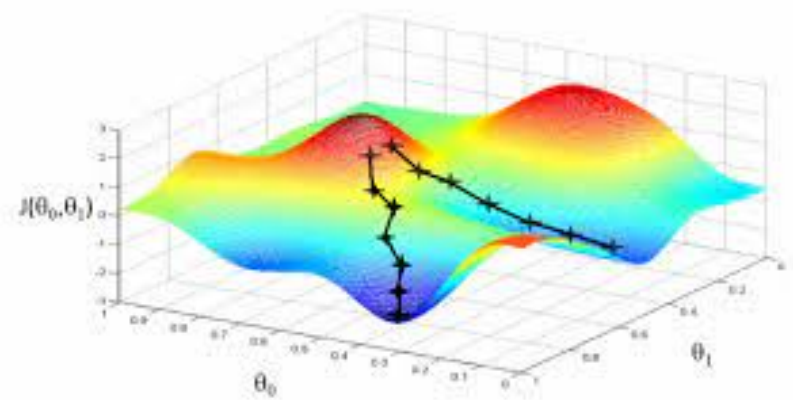
IF I CHANGE A AND B A LITTLE BIT, DOES THE LOSS CHANGE A LITTLE OR A LOT?

KEY INSIGHT: LOSS IS *DIFFERENTIABLE*

IN 1D: DERIVATIVE TELLS US HOW MUCH THE FUNCTION CHANGES LOCALLY



IN 2D+: DERIVATIVE TELLS US HOW MUCH THE FUNCTION CHANGES LOCALLY AND WHICH DIRECTION IT CHANGES THE FASTEST



TO DECREASE LOSS, WE CAN FOLLOW THE GRADIENT!

Example: We tried a = 0, b = 5.

The (negative) gradient of the function at these values is (39320, 384), so we would increase both a and b.

Keep doing this, and eventually we'll arrive at the correct values for a and b.

MACHINE LEARNING, OVERALL SETUP

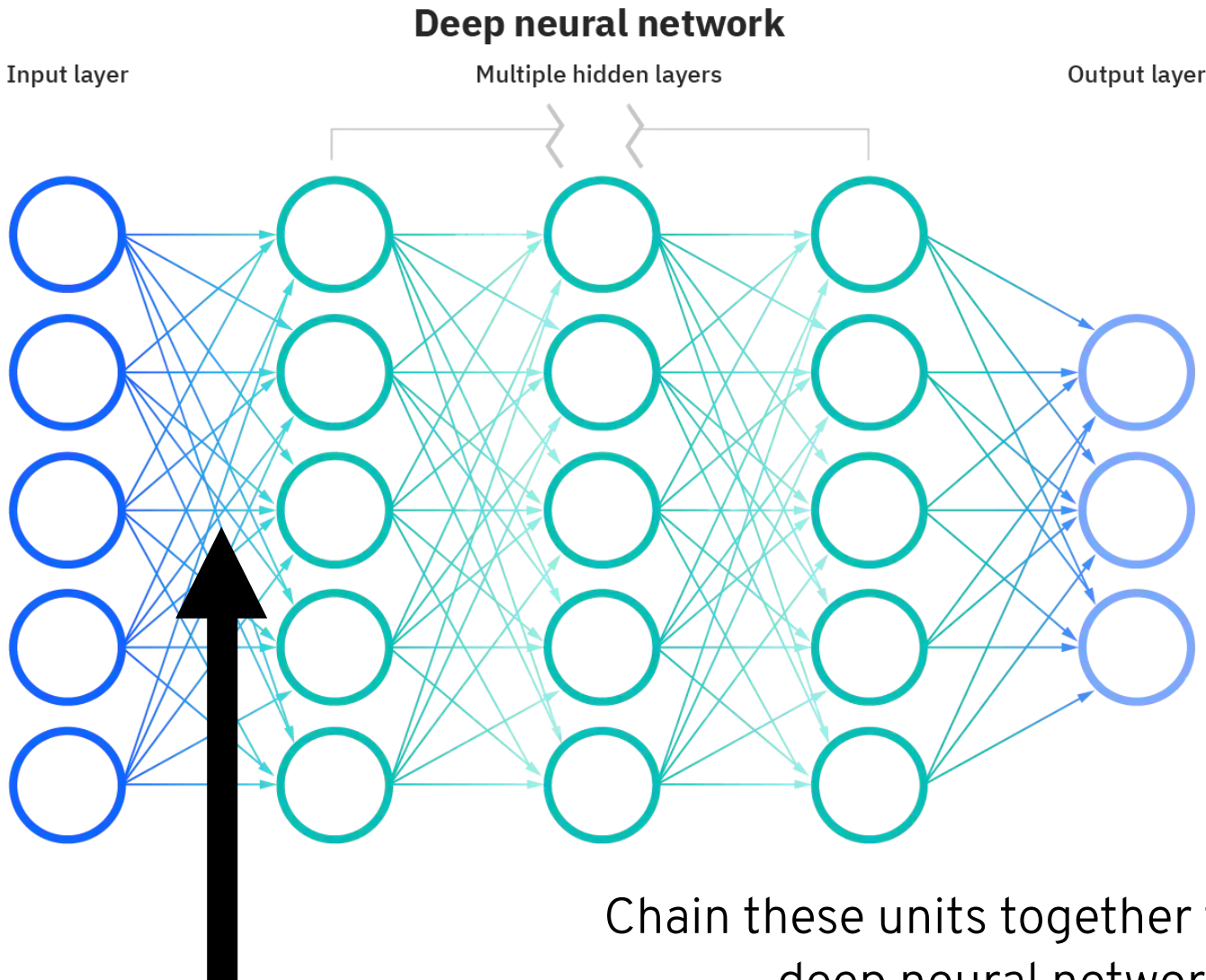
1. Define a function we want to compute, with some *parameters* that affect the function's output.
2. Get *examples* of the function we want to compute
3. Define a *loss function*, which defines how much *our* function didn't match the examples
4. Compute the *gradient* the loss with respect to the parameters.
5. Update the parameters according to the gradient.
6. Repeat until the function does more-or-less what we want.

NEURAL NETWORKS AND DEEP LEARNING

IN THE 2000S, PEOPLE REALIZED THAT THE FOLLOWING OPERATION MAKES FOR A PRETTY GOOD BASIC UNIT FOR A MACHINE LEARNER

$$y = relu(Wx + b)$$

Where x is an input vector, W is a weight matrix, b is a bias vector, y is the output, and relu is a ReLU operation.



Chain these units together to make a deep neural network!

$$y = relu(Wx + b)$$

HOW MANY PARAMETERS?

$$N^2 + N$$ ***PER-LAYER***

In large neural networks, this adds up to *millions* of parameters to learn!

This is very expensive to train.

Use specialized hardware to train these learners!

GPUs are very good at doing the kinds of calculations that neural nets need for training.

CONCEPT DEMO

https://www.tensorflow.org/tutorials/quickstart/beginner

CONTINUING PROGRAMMING

This course has covered the most basic principles of programming.

If you want to keep learning, you have lots of options:

- Take an intro to programming course at your college. These will cover lots of similar subjects to what we did in this class, but they will also touch on some different things.
- Take a data structures course. These have a million different names, but they are almost always the course right after the introductory programming course.
- Online options like https://exercism.org/

GENERAL ADVICE

When focusing on jobs or abilities, people always focus on languages and libraries (e.g. MATLAB, R, Flutter, React, etc.)

"Need someone with N years of Python", "Experienced in Angular", etc. etc.

If you want to work as a software developer, this is important.

IF YOU JUST WANT TO BE ABLE TO WRITE PROGRAMS, WORRY ABOUT WRITING PROGRAMS FIRST! WORRY ABOUT LANGUAGES LATER.