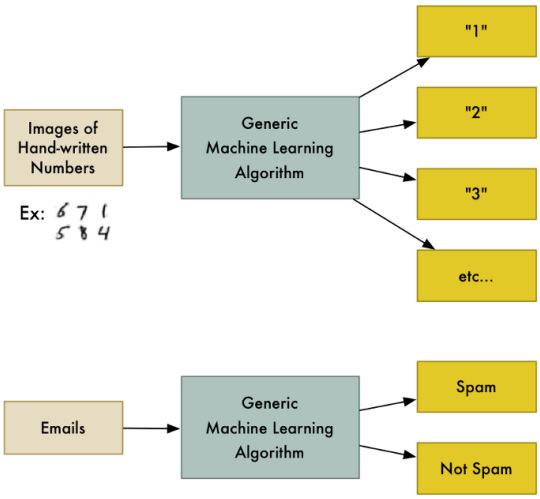
What is machine learning?

Machine learning is the idea that there are generic algorithms that can tell you something interesting about a set of data without having to write any custom code specific to the problem. Instead of writing code, you feed data to the generic algorithm and it builds its own logic based on the data.

One kind of algorithm is a classification algorithm. It puts data into different groups. The same code that can recognize handwritten numbers could also be used to classify emails into span and not-spam without changing anything. Same algorithm but with different training data.



1 ML algorithm

Supervised Learning

Assume I am a real estate agent. My business is growing, so I hire a bunch of new trainee agents to help you out. But there's a problem — I can glance at a house and have a pretty good idea of what a house is worth, but my trainees don't have my experience so they don't know how to price their houses.

To help the trainees, I write a little app that can estimate the value of a house in my area based on it's size, neighborhood, etc, and what similar houses have sold for.

So I write down every time someone sells a house in the city for 3 months. For each house, I write down a bunch of details — number of bedrooms, size in square feet, neighborhood, etc. But most importantly, I write down the final sale price:

Bedrooms	Sq. feet	Neighborhood	Sale price
3	2000	Normaltown	$250,000
2	800	Hipsterton	$300,000
2	850	Normaltown	$150,000
1	550	Normaltown	$78,000
4	2000	Skid Row	$150,000

Using that training data, we want to create a program that can estimate how much any other house in my area is worth:

Bedrooms	Sq. feet	Neighborhood	Sale price
3	2000	Hipsterton	???

This is called supervised learning. I knew how much each house sold for, so in other words, I knew the answer to the problem and could work backwards from there to figure out the logic.

This is called supervised learning. I knew how much each house sold for, so in other words, I knew the answer to the problem and could work backwards from there to figure out the logic.

This kind of like having the answer key to a math test with all the arithmetic symbols erased:

Math Quiz #1 - Teacher's Answer Key

1) 2 4 5 = 3	5) 6 2 2 = 10
2) 5 2 8 = 2	6) 3 1 1 = 2
3) 2 2 1 = 3	7) 5 3 4 = 11
4) 4 2 2 = 6	8) 1 8 1 = 7

From this, can I figure out what kind of math problems were on the test? I know I am supposed to "do something" with the numbers on the left to get each answer on the right.
In supervised learning, I am letting the computer work out that relationship for me. And once I know what math was required to solve this specific set of problems, I could answer to any other problem of the same type!

Pseudocode Assuming no ML

```
def estimate_house_sales_price(num_of_bedrooms, sqft, neighborhood):
  price = 0
  # In my area, the average house costs $200 per sqft
  price_per_sqft = 200
  if neighborhood == "hipsterton":
    # but some areas cost a bit more
    price_per_sqft = 400
  elif neighborhood == "skid row":
    # and some areas cost less
    price_per_sqft = 100
  # start with a base price estimate based on how big the place is
  price = price_per_sqft * sqft
  # now adjust our estimate based on the number of bedrooms
  if num_of_bedrooms == 0:
    # Studio apartments are cheap
    price = price - 20000
  else:
    # places with more bedrooms are usually
    # more valuable
    price = price + (num_of_bedrooms * 1000)
  return price
```

Wouldn't it be better if the computer could just figure out how to implement this function for me? Who cares what exactly the function does as long as it returns the correct number:

```
def estimate_house_sales_price(num_of_bedrooms, sqft, neighborhood):
  price = <computer, plz do some math for me>
  return price
```

One way to think about this problem is that the **price** is a delicious stew and the ingredients are the **number of bedrooms**, the **square footage** and the **neighborhood**. If I could just figure out how much each ingredient impacts the final price, maybe there's an exact ratio of ingredients to stir in to make the final price. That would reduce the original function (with all those crazy if's and else's) down to something really simple like this:

```
def estimate_house_sales_price(num_of_bedrooms, sqft, neighborhood):
 price = 0
 # a little pinch of this
 price += num_of_bedrooms * .841231951398213
 # and a big pinch of that
 price += sqft * 1231.1231231
 # maybe a handful of this
 price += neighborhood * 2.3242341421
 # and finally, just a little extra salt for good measure
 price += 201.23432095
 return price
```

The numbers are our **weights**, if we can figure out the perfect weights then our function could predict house prices.

Simple way to figure out weights:

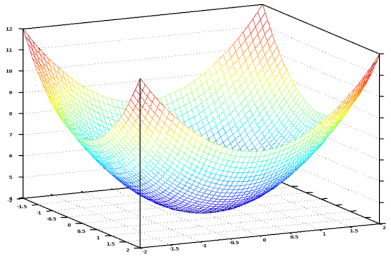
1. Start with each weight at 1.0
2. Run through training set and see how wrong it is using cost function
3. Repeat with a ton of weights

1. Start with each weight at 1.0
2. Run through training set and see how wrong it is using cost function
3. Repeat with a ton of weights

Cost Function example:

$$J(\theta) = \frac{1}{2m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)} \right)^2$$

Example of the Graph of Cost Function:



Batch Gradient Descent Image