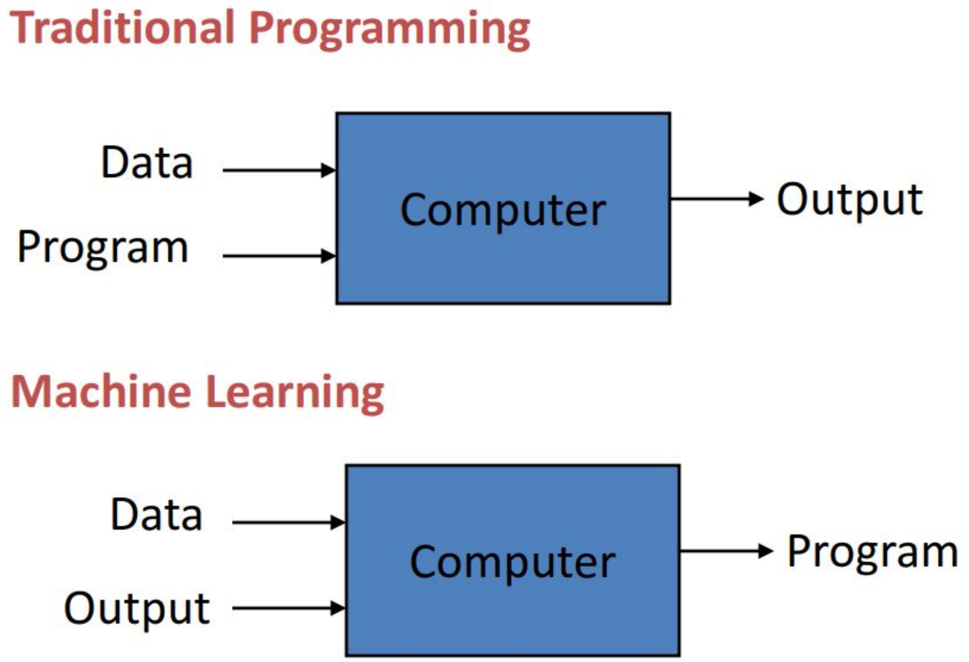
INTRODUCTION TO MACHINE LEARNING

Jackson Hassell

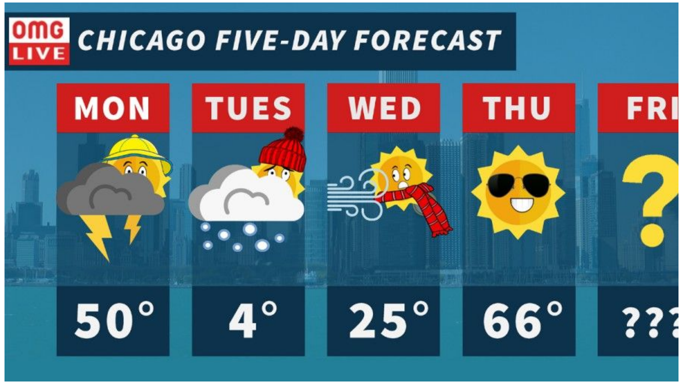
What is machine learning?



Instead of the programmer manually writing an algorithm, the model learns from the data by itself.

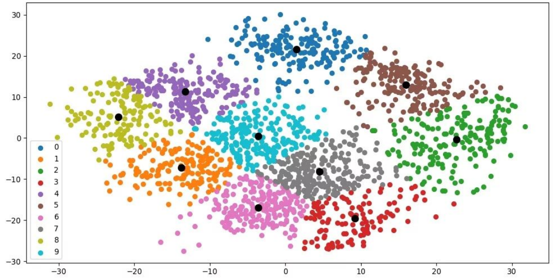
The end result is always a model that, given a certain input, returns a prediction.

What is the model predicting?



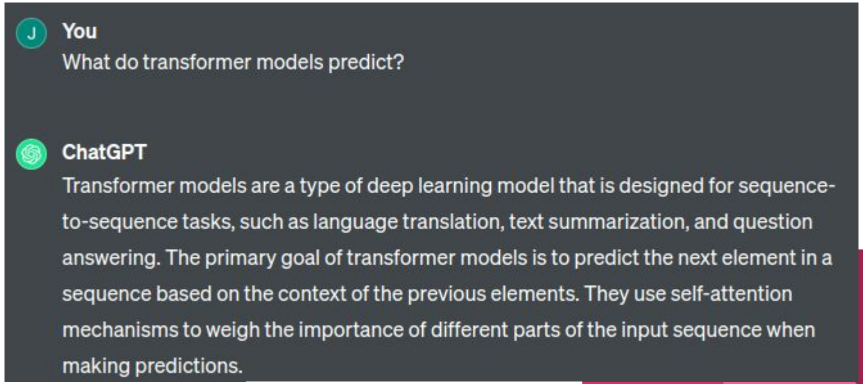
Next day's weather

Objects in an image



Cluster labels from unlabeled data

How to move each "muscle" to walk

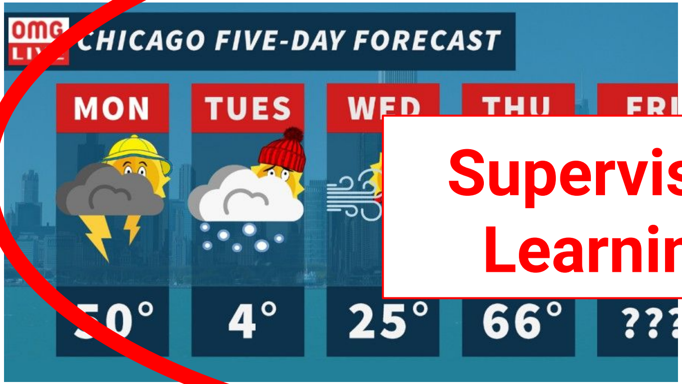


You
What do transformer models predict?

ChatGPT
Transformer models are a type of deep learning model that is designed for sequence-to-sequence tasks, such as language translation, text summarization, and question answering. The primary goal of transformer models is to predict the next element in a sequence based on the context of the previous elements. They use self-attention mechanisms to weigh the importance of different parts of the input sequence when making predictions.

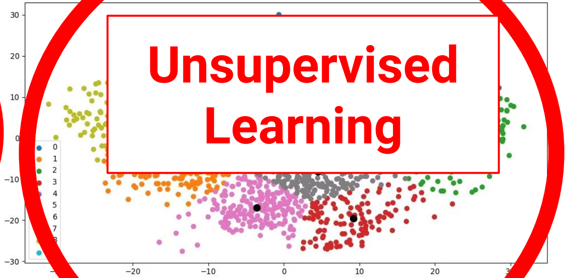
Next word in a sentence

What is the model predicting?



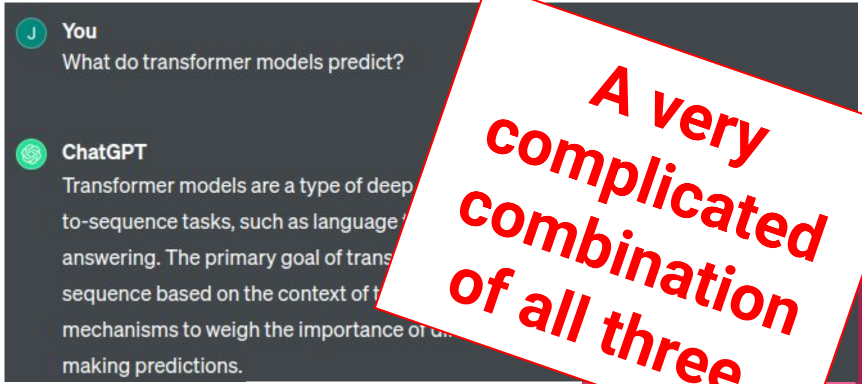
Next day's weather

Objects in an image



Cluster labels from unlabeled data

How to move each "muscle" to walk



You
What do transformer models predict?

ChatGPT
Transformer models are a type of deep
to-sequence tasks, such as language
answering. The primary goal of trans
sequence based on the context of t
mechanisms to weigh the importance of
making predictions.

Next word in a sequence

Supervised Learning

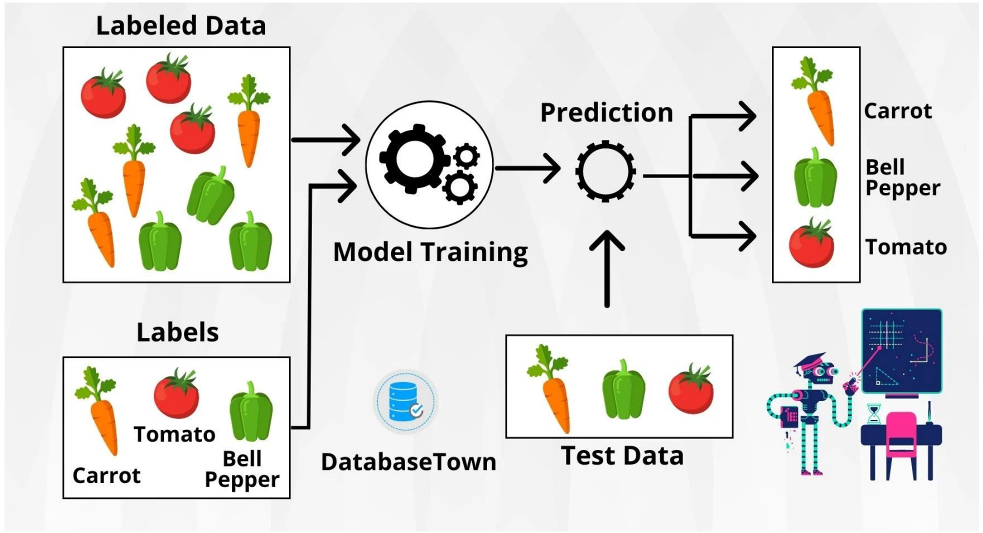
Unsupervised Learning

Reinforcement Learning

A very complicated combination of all three.

Supervised Learning

Supervised Learning

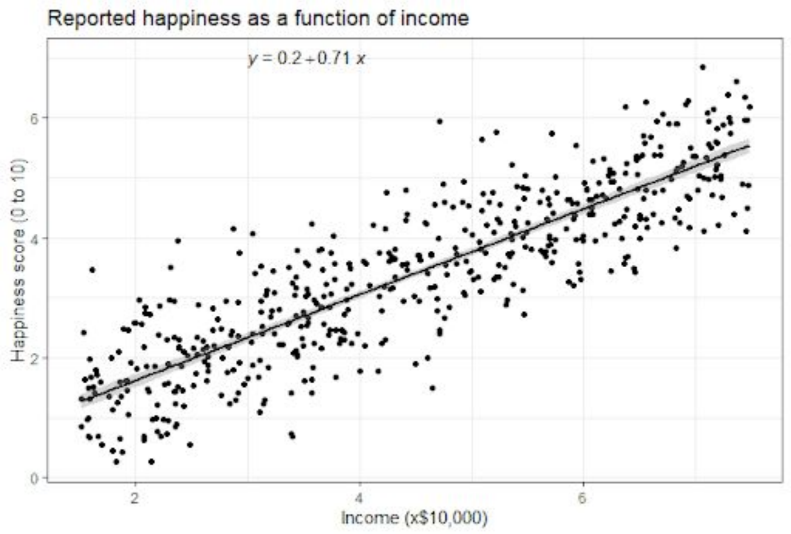


The model learns round + red = tomato just from labeled examples.

- Given a set of inputs and outputs, learn to predict the output from the input.
    - Each kind of supervised model learns the associations between input and output differently. But at the end of the day, they all create a mapping between input and output, and you can treat them as a black box
- The more complex the problem, and the more complex the model, the more data you need.
    - Getting the outputs to train on is the hard part. Many tasks require humans to label millions of data points.

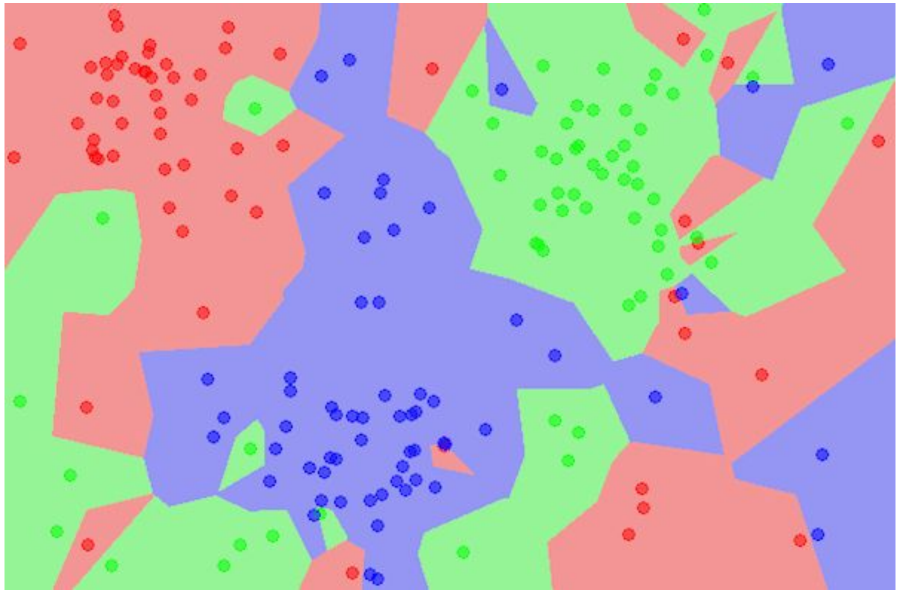
Linear Regression

- The most basic supervised model out there.
- Draws a “line of best fit.”
    - In an X-Y plane, that means solving for m and b in y = m*x + b.
    - For multidimensional data, this looks like y = m1*x1 + m2*x2 + m3*x3 + b.
- Benefit: easily interpretable.
    - Each learned weight is easy to understand. It means, for every time x increases by 1, y increases by m.
- Drawback: can only represent linear relationships between input and output.
    - If you have non-linear relationships in your data, you either need to transform them so that they’re linear, or use a different model altogether.



K-Nearest Neighbors

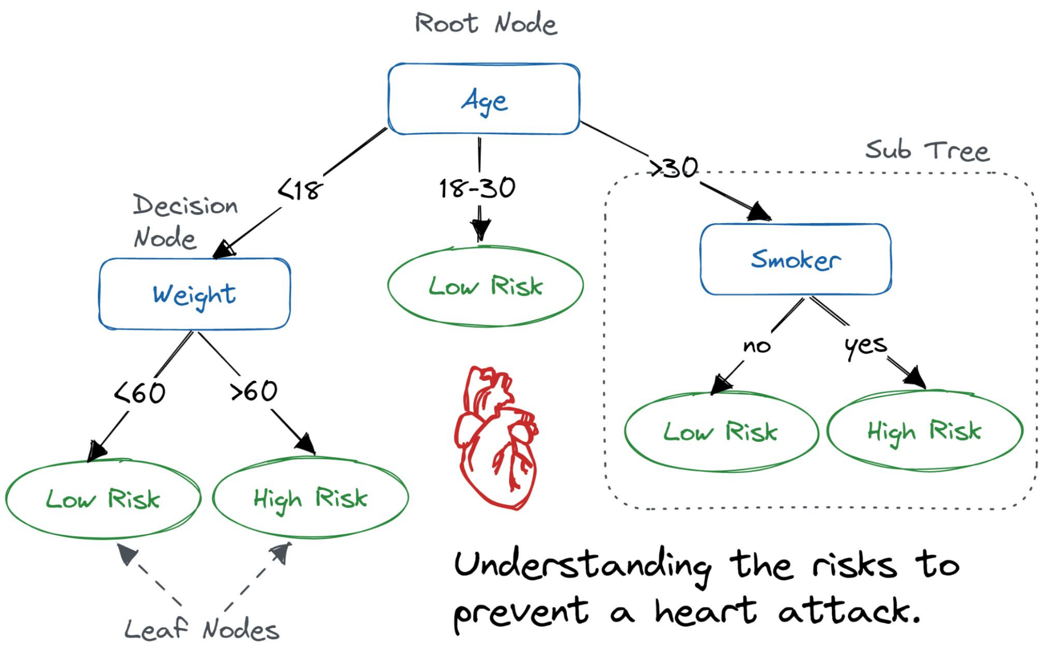
- One of the simplest models, but it still manages to outperform large neural networks on many tasks.
- Creates a map showing how close each input data point is to each other. At prediction time, simply find the closest K data points, average their output values, and return it.
    - K is a hyperparameter.
    - Can use different definitions of "closest."
- Benefits:
    - Versatile and easy to understand.
- Drawbacks:
    - Computationally and memory inefficient.
        - Representing a training database of 1 million datapoints with KNN is slow and takes a good amount of computing power.



k=1

Decision Tree

- Sequentially splits the data into two or more branches at each node.
  - It does this based on how well it separates your classes.
- At prediction time, simply follow the tree from the root node to the leafs.
- How many branches per split, and how many splits in a tree are hyperparameters.
- Benefits:
  - Highly interpretable and can represent complex relationships.
- Drawbacks:
  - Needs extensive hyperparameter tuning. Too few splits and the model is useless. Too many and it simply memorizes the training data and won't generalize well.
- Decision trees can be ensembled in lots of fun ways.
  - What would happen if you made 10 decision trees, gave each one access to only 20% of your inputs, and averaged their predictions? A new kind of model called a Random Forest.
  - Or what if you trained a second decision tree to predict how wrong the first tree is, then adjust the original prediction? That's XGBoost (mostly).



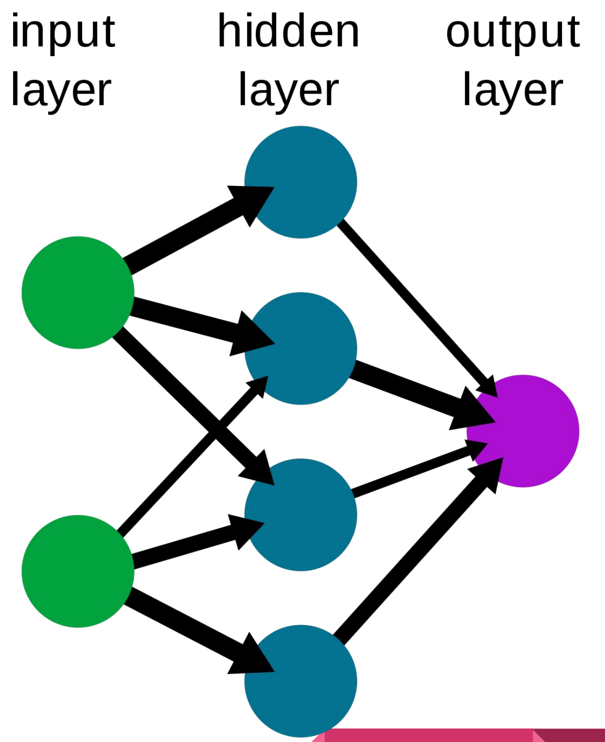
Understanding the risks to prevent a heart attack.

Neural Networks

Neural Networks

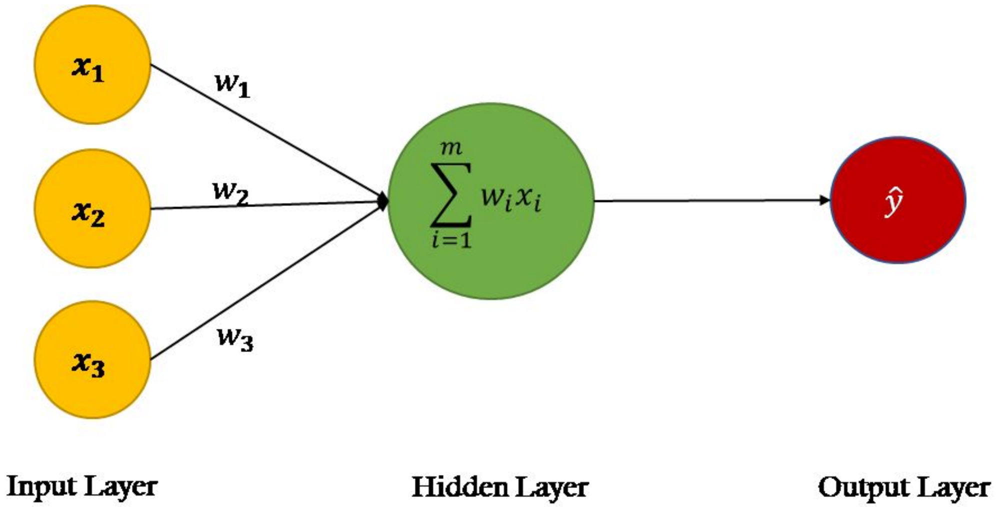
- The backbone of modern machine learning.
  - If you hear "deep learning" it means using a large neural network.
- Made of hundreds of "neurons" spread across multiple layers.
- The first layer takes the training data as input, and the output of the last layer is the model's prediction.
  - What happens in the middle gets weird.

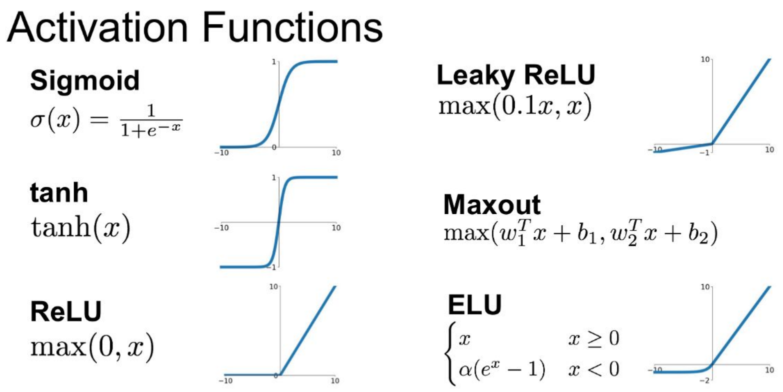
A simple neural network



Anatomy of a Neuron

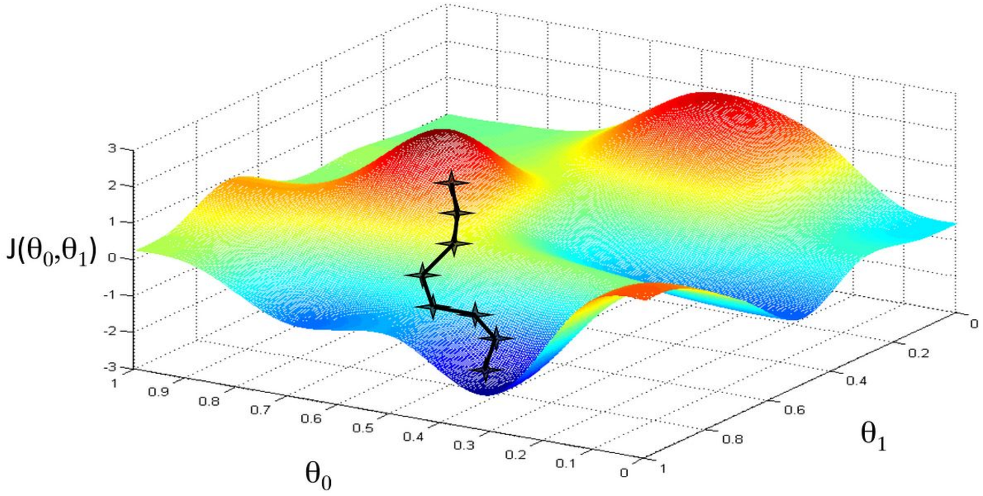
- Neural network neurons were designed to mimic the behavior of biological neurons, but really they're just "linear" regressors.
- Each neuron receives as input the output of each neuron in the previous layer.
- Each input is multiplied by a weight, then summed together. This sum is then passed through an activation function before being sent to the next layer.
    - A non-linear activation function must be used because otherwise the entire neural network effectively collapses into a big linear regressor itself.
- The values for these weights are what the model learns when it's given the training data. But how does it know what value of weight is correct?





Training Neural Networks

- Initially, all neuron weights are randomly set. This leads to a model with a very high error rate at first.
- Using the power of calculus, you can calculate the gradient of the model's error with respect to the weights.
- You then update the weights using this gradient, and repeat the process until the error is low enough*.
- This is called gradient descent.

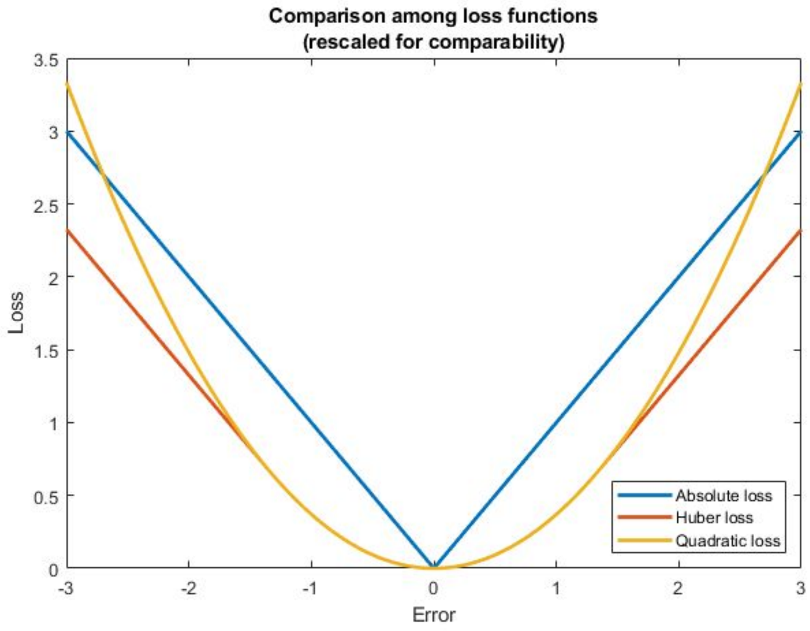


*There are a lot of extra factors that go into this, like how much you descend at each step, how you avoid getting stuck in local minima, how you seed your initial weights, how you know when to stop, etc.

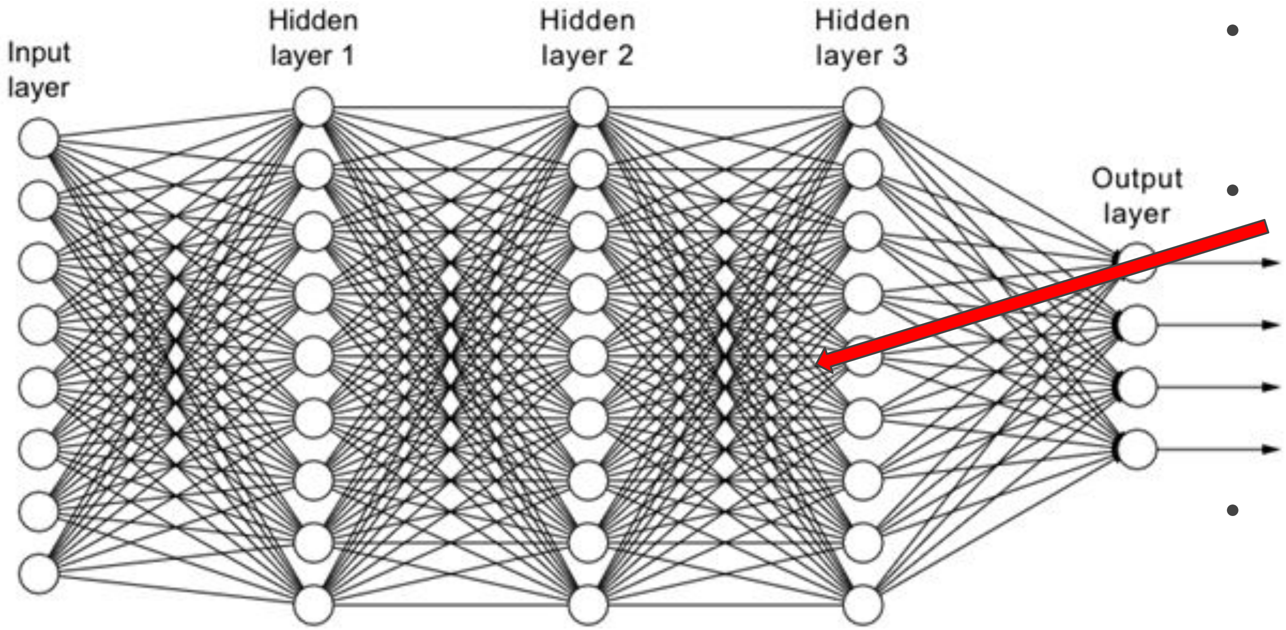
THIS IS YOUR MACHINE LEARNING SYSTEM?
YUP! YOU POUR THE DATA INTO THIS BIG PILE OF LINEAR ALGEBRA, THEN COLLECT THE ANSWERS ON THE OTHER SIDE.
WHAT IF THE ANSWERS ARE WRONG?
JUST STIR THE PILE UNTIL THEY START LOOKING RIGHT.
DATA
ANSWERS

Loss Functions

- Should we calculate the gradient of the weights with respect to the raw error, or do something more?
- Loss is a function you pass your error into. You then calculate the gradient of the loss with respect to the weights instead.
  - This unlocks all sorts of new behaviors for neural networks.
  - You can think of different loss functions as different ways of "stirring" your pile of linear algebra.
- How do you choose your loss function?
  - Do you want to minimize false positives or false negatives or do you care just about overall accuracy? Are you doing classification or regression? How tolerant are you of frequent small errors vs occasional large errors?
  - The loss function you choose changes based on what you need from the model.



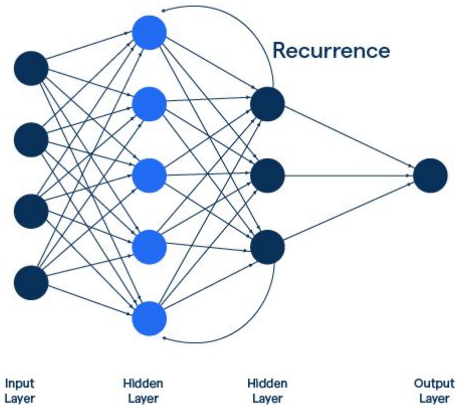
These networks can get very large very quickly.

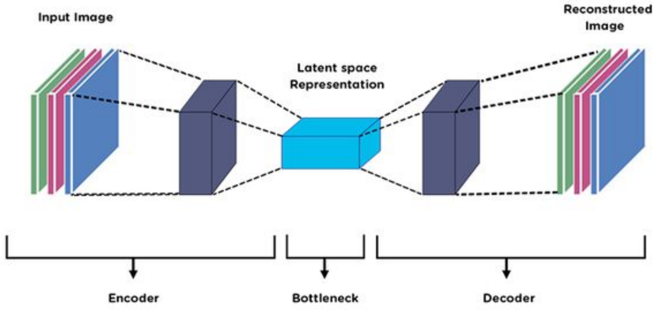


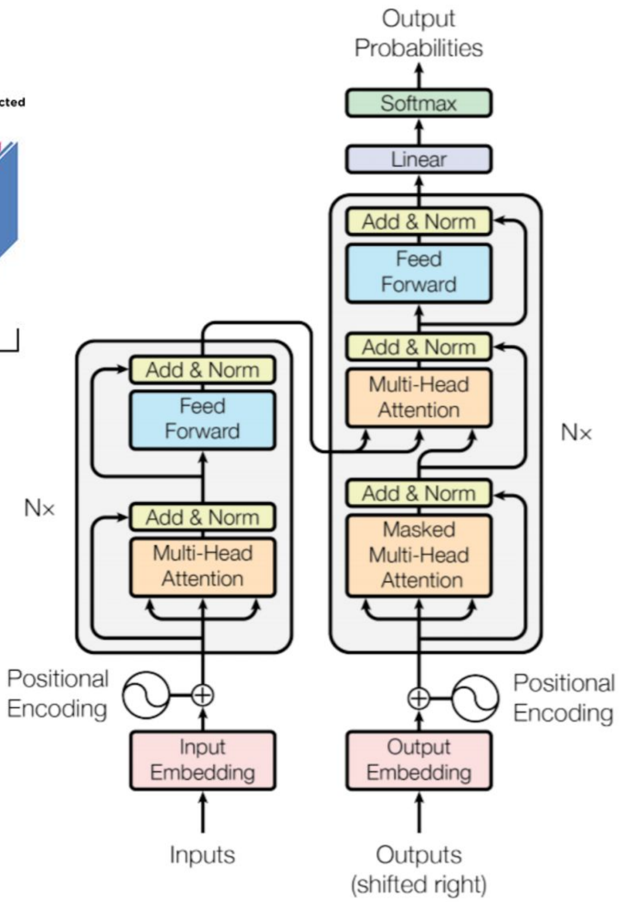
- You may have heard the phrase “No one understands how deep learning understands things.”
- How would you interpret the weights here?
  - Called a latent space transformation.
  - There is no known way to extract what a neural network has learned from its weights alone.
- Neural networks are treated as “black boxes.”

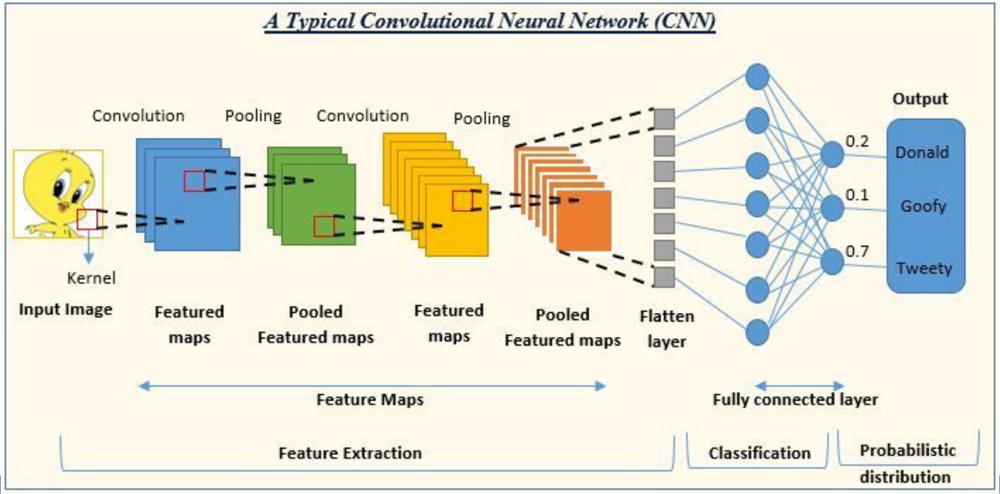
- The “deep” part in deep learning refers to neural networks with many hidden layers.
  - The neural network visualized above wouldn’t be considered deep - 40 neurons total is tiny, relatively speaking.
  - Something this small could be trained easily on a cheap laptop.
- A common way of representing a network’s size is by its parameter count - that is, how many weights it has to learn.
  - GPT-4 - the current best language model - has 1.7 trillion parameters.
  - GPT-4 cost over $100M dollars of supercomputer time to train.

Recurrent Neural Network





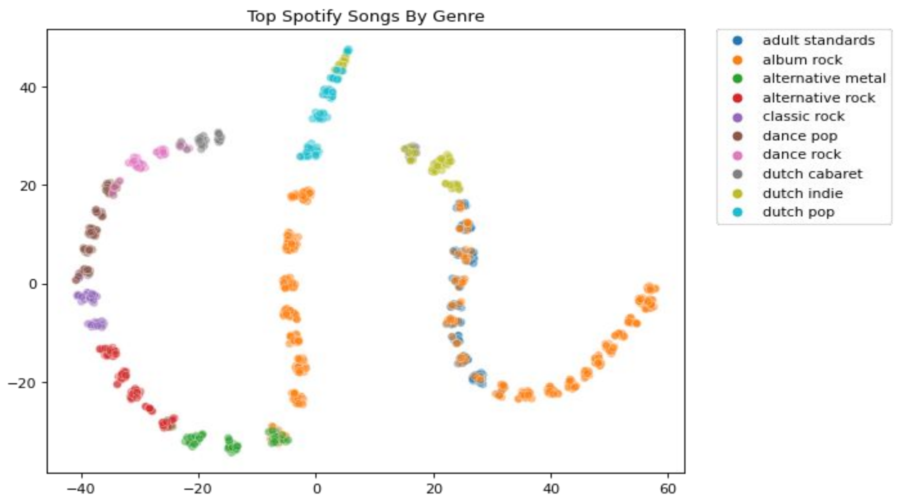




Unsupervised Learning

Unsupervised Learning

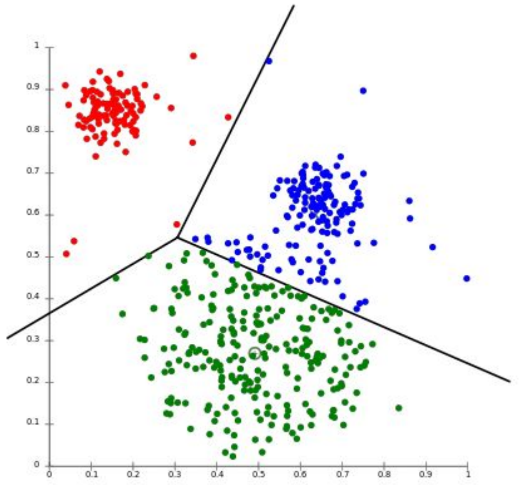
- What do you do if your data doesn't have any labels?
- Ways of finding structure in unlabeled data is called unsupervised learning.
- This includes:
  - Association rules.
    - How Amazon populates that list of "Customers who bought this item bought these items too."
  - Clustering:
    - What features tend to go together?
    - Are some data points more similar to each other than others?
  - Dimensionality reduction:
    - How do you plot 5-dimensional data on an X-Y graph?
  - Anomaly detection.
  - And more!



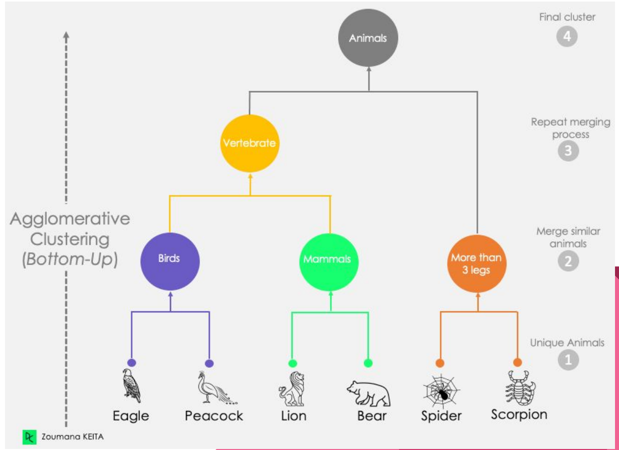
Dimensionality reduction is great for visualizations.

Clustering

- Groups similar data points together.
- K-means clustering:
  - Randomly assigns K data points as “centroids”.
    - K is a hyperparameter.
  - Assigns each data point to the nearest centroid.
    - Can use different definitions of “nearest.”
  - Calculate the average position of each cluster, and have that be the new centroid.
  - Assign each data point to the nearest centroid.
  - Repeat until no data points changed cluster during assignment.
- Hierarchical clustering:
  - Each data point starts as its own cluster.
  - Merge the two closest clusters together.
  - Repeat until all data points belong to the same cluster.
  - Great for representing the relationship between different clusters, but less straightforward to interpret than k-means.

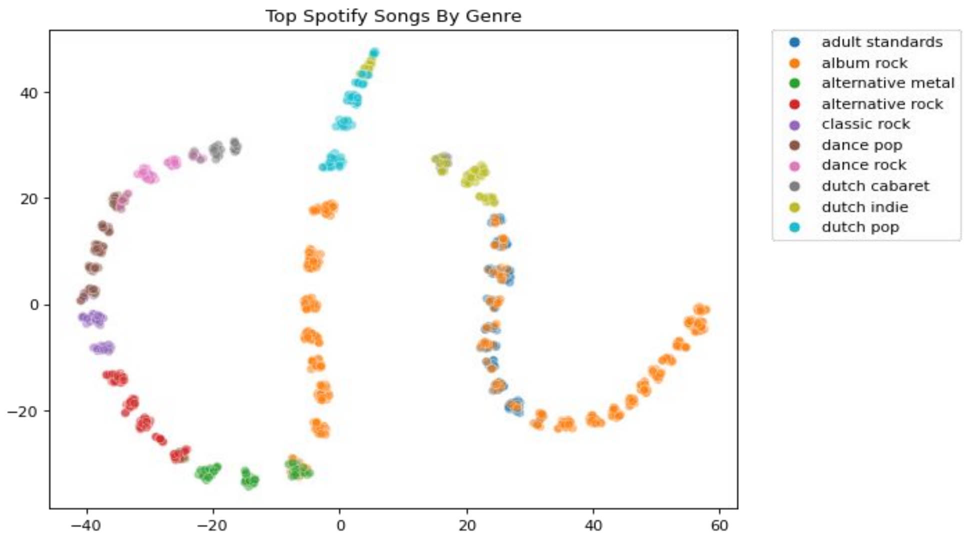


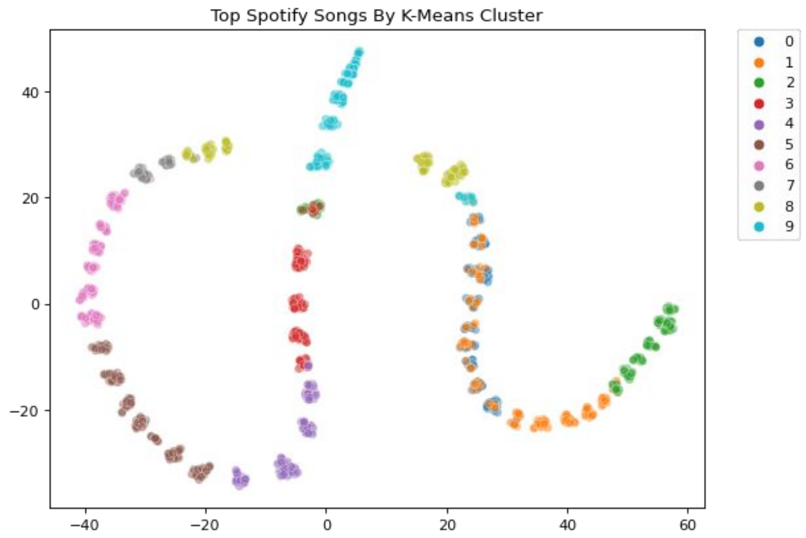
K-means clustering



Hierarchical Clustering

Clustering can be almost as useful as supervised classification models, without the need to painstakingly label the data.

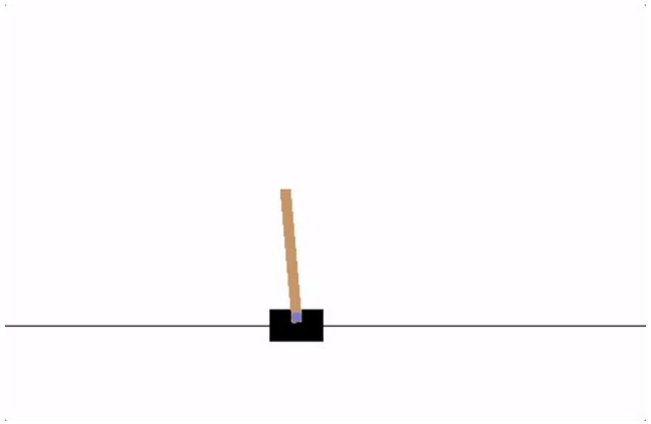




Reinforcement Learning

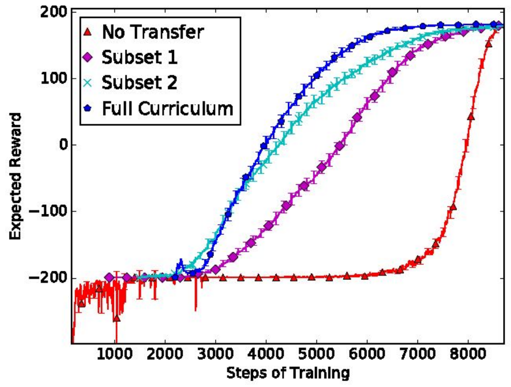
Reinforcement Learning

- Requires:
    - Model that can predict actions.
    - Environment to act in.
    - Reward function that grades each action.
- At first, the model makes random actions in the environment.
- The reward function grades each action.
    - Stomping Goomba = +5 points.
    - Touching spikes = -15 points.
- The model eventually makes actions that results in high rewards through random chance. It then learns to take actions that result in high rewards.
    - This can take millions of repetitions before the model significantly improves, depending on the complexity of the environment and actions.



Reinforcement Learning is Complicated

- RL is conceptually simple, but complex in implementation.
- How does the model learn to predict which action results in high rewards?
    - Supervised learning! But there's a whole set of neural network architectures for RL specifically.
- What about actions where you get no rewards immediately, but lead to high rewards later on?
    - Smear rewards over many actions, weighted towards actions closer to the rewarded frame.
- How do you decide how many reward points each action should get?
    - No one knows. Try a few different reward schemes to see which does the best.
- How do you format the environment as input to the model?
    - Also a hyperparameter.
- RL is very slow to train - millions of simulations take time.
    - The more complex the environment, actions, and model are, the more training steps it will take.



It can take a long time before the model starts making intelligent actions. Sometimes it just never converges.

MARIO
000000
×00
WORLD
1-1
TIME
399

Reinforcement Learning is Powerful

- AlphaGo (an RL model for the game Go) beat the best Go player in the world in 2017.
- Waymo - a self-driving cab company that uses RL models to drive the cars - is already operating in several cities, with much lower accident rates than humans.
- Google created an RL model to control cooling in its data centers, resulting in a 40% reduction in energy use.
- Human-like robots (run by RL models) from Boston Dynamics are able to walk around and traverse terrain by themselves.

Thank You