Foundation Models

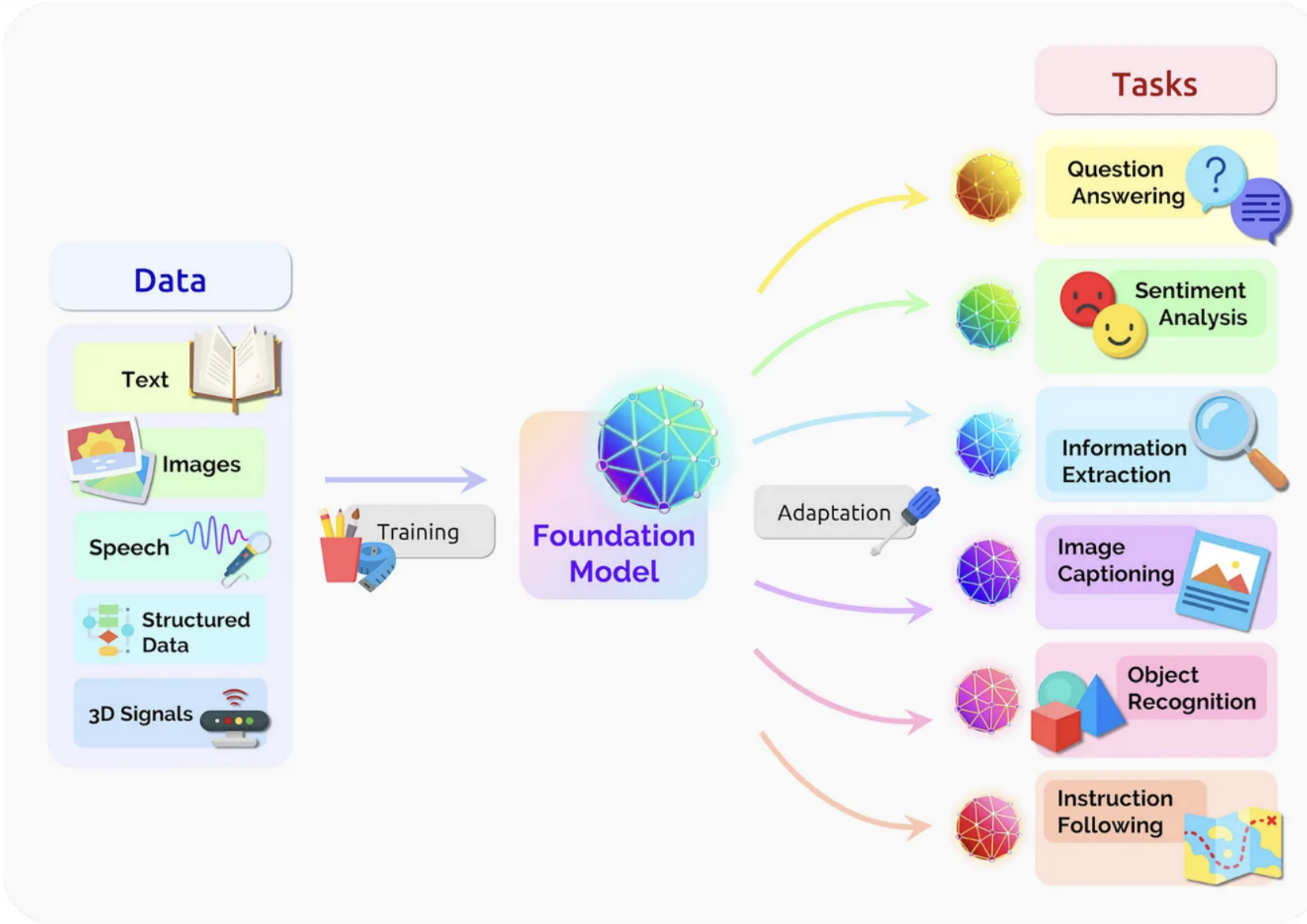
based on: https://writings.stephenwolfram.com/2023/02/what-is-chatgpt-doing-and-why-does-it-work/

Goals

- What is a “Foundation Model”?
- How do they work?
- Why would you want to do this?
- Point out resources for working code.

What is a Foundation Model?

- Large AI model trained on vast amounts of unlabeled data
- Implements self-supervised learning
- Can be adapted to specific tasks
- Exhibits *emergence* and *homogenization*


Data
Text
Images
Speech
Structured Data
3D Signals
Training
Foundation Model
Adaptation
Tasks
Question Answering
Sentiment Analysis
Information Extraction
Image Captioning
Object Recognition
Instruction Following

These models have a lot of parameters and are trained on vasts amounts of data

Year	Model	# of Parameters	Dataset Size
2019	BERT [39]	3.4E+08	16GB
2019	DistilBERT [113]	6.60E+07	16GB
2019	ALBERT [70]	2.23E+08	16GB
2019	XLNet (Large) [150]	3.40E+08	126GB
2020	ERNIE-Gen (Large) [145]	3.40E+08	16GB
2019	RoBERTa (Large) [74]	3.55E+08	161GB
2019	MegatronLM [122]	8.30E+09	174GB
2020	T5-11B [107]	1.10E+10	745GB
2020	T-NLG [112]	1.70E+10	174GB
2020	GPT-3 [25]	1.75E+11	570GB
2020	GShard [73]	6.00E+11	–
2021	Switch-C [43]	1.57E+12	745GB

LLM's are basically trying to predict the next word . . .

The best thing about AI is its ability to

learn	4.5%
predict	3.5%
make	3.2%
understand	3.1%
do	2.9%

- We can illustrate this using a pre-trained neural net, GPT-2, implemented easily using *Mathematica*.

- We'll get to how this neural net works in a few slides.

```
model = NetModel[{"GPT2 Transformer Trained on WebText Data",
    "Task" -> "LanguageModeling"}]
```

- Let's give it a sentence and see how it does...

```
model["The best thing about AI is its ability to",
{"TopProbabilities", 5}]
```

Out[9]//TableForm=

do	0.028852
understand	0.0307757
make	0.031908
predict	0.0349726
learn	0.0445358

- We can ask it to continue completing the setence...

```
NestList[
 StringJoin[#,
   model[#,
    "Decision"]] &,
    "The best thing about AI is its ability to", 7]
```

- And it gives us the following output...

- Ok, but probably not the best. Becomes "flat" very quickly.

The best thing about AI is its ability to learn from experience. It's not just a matter of learning from experience, it's learning from the world around you. The AI is a very good example of this. It's a very good example of how to use AI to improve your life. It's a very good example of how to use AI to improve your life. The AI is a very good example of how to use AI to improve your life. It's a very good example of how to use AI to

- Researchers found that they could get more realistic results by adding a random component to the model (i.e. don't always go with the highest probability word.

The best thing about AI is its ability to see through, and make sense of, the world around us, rather than panicking and ignoring. This is known as AI "doing its job" or AI "run-of-the-mill." Indeed, taking an infinite number of steps, developing a machine that can be integrated with other systems, or controlling one system that's truly a machine, is one of the most fundamental processes of AI. Aside from the human-machine interaction, AI was also a big part of creativity

- When implemented with GPT-3, which was trained on more data, the results are even better...

The best thing about AI is its ability to learn and develop over time, allowing it to continually improve its performance and be more efficient at tasks. AI can also be used to automate mundane tasks, allowing humans to focus on more important tasks. AI can also be used to make decisions and provide insights that would otherwise be impossible for humans to figure out.

Where do the probabilities come from?

- We can pull some data from wikipedia, like the counts of letters in the article on cats and dogs...

```
# Just cats
LetterCounts[WikipediaData["cats"]]

#Just dogs
LetterCounts[WikipediaData["dogs"]]

# All of English
Entity["Language",
 "English::385w8"]
 [EntityProperty["Language", "LetterFrequency"]]
```

- We begin by generating a sequence of letters using the probabilities above.
- Unsurprisingly, this is nonsense.

rronoitadatcaeaesaotdoysaroiyiinnbantoioestlhddeocneooewceseciselnodrtrdgriscsatsepesdcnio
uhoetsedeyhedslernevstothindtbmnaohngotannbthrdthtonsipieldn

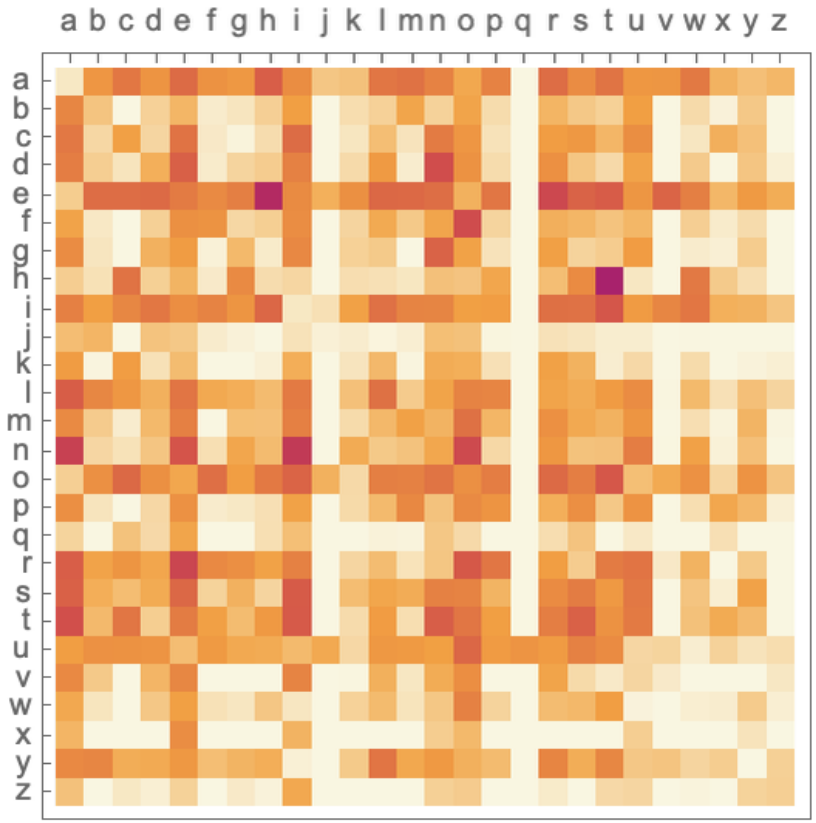
- We can do a bit better by putting in spaces and forcing the word lengths to follow the same probability distribution as in English.

- But not much better...

ni hilwhuei kjtn isjd erogofnr n rwhwfao rcuw lis fahte uss cpnc

nlu oe nusaetat llfo oeme rrhrtn xdses ohm oa tne ebedcon oarvthv ist

- What about looking at 2-grams?



- We generate out letters two at a time.
- And get this...

on inguman men ise forerenoft weat iofobato buc ous corew ousesetiv

falle tinouco ryefo ra the ecederi pasuthrgr cuconom tra tesla wil tat pere thi

- What if we start using 3-grams, 4-grams, or 5-grams...

0	on gxeeetowmt tsifhy ah aufnsoc ior oia itlt bnc tu ih uls
1	ri io os ot timumumoi gymyestit ate bshe abol viowr wotybeat mecho
2	wore hi usinallistin hia ale warou pothe of premetra bect upo pr
3	qual musin was witherins wil por vie surgedygua was suchinguary outheydays theresist
4	stud made yello adenced through theirs from cent intous wherefo proteined screa
5	special average vocab consumer market prepara injury trade consa usually speci utility

- Now let's do what chat GPT does, use whole words.
- There are about 40,000 commonly used words in English.
- If we look at a really large corpus, we can get an idea of their probabilities.
- Here's an example sentence using 1-grams...

of program excessive been by was research rate not here of of other is men
were against are show they the different the half the the in any were leaved

- Here's what we get with bigrams...

cat through shipping variety is made the aid emergency can the

cat for the book flip was generally decided to design of

cat at safety to contain the vicinity coupled between electric public

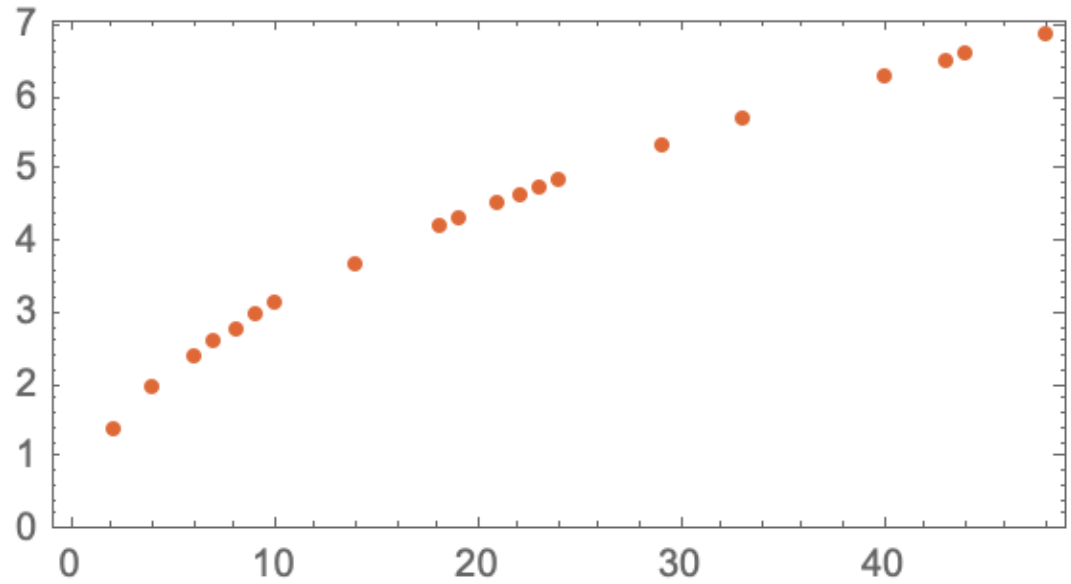
cat throughout in a confirmation procedure and two were difficult music

cat on the theory an already from a representation before a

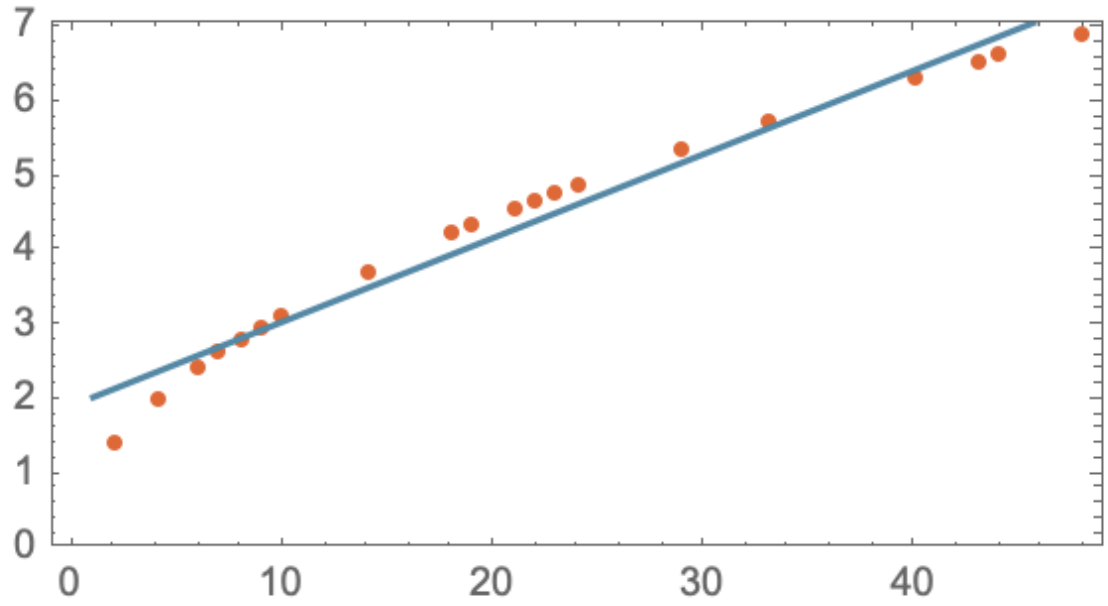
- But, we can't do with words what we were doing with individual letters.
- By crawling the web and using all the digitized books on the net, we might get 500 billion or so words of text to build our probabilities with.
- With 40,000 common words, already the possible 2-grams is 1.6 billion.
- With 3-grams it is 60 trillion.
- So there are, literally, not enough data out there for us to get the probabilities down.
- We need a model...

- Imagine you're trying to figure out how long it takes a cannon ball to drop out various windows to the ground.

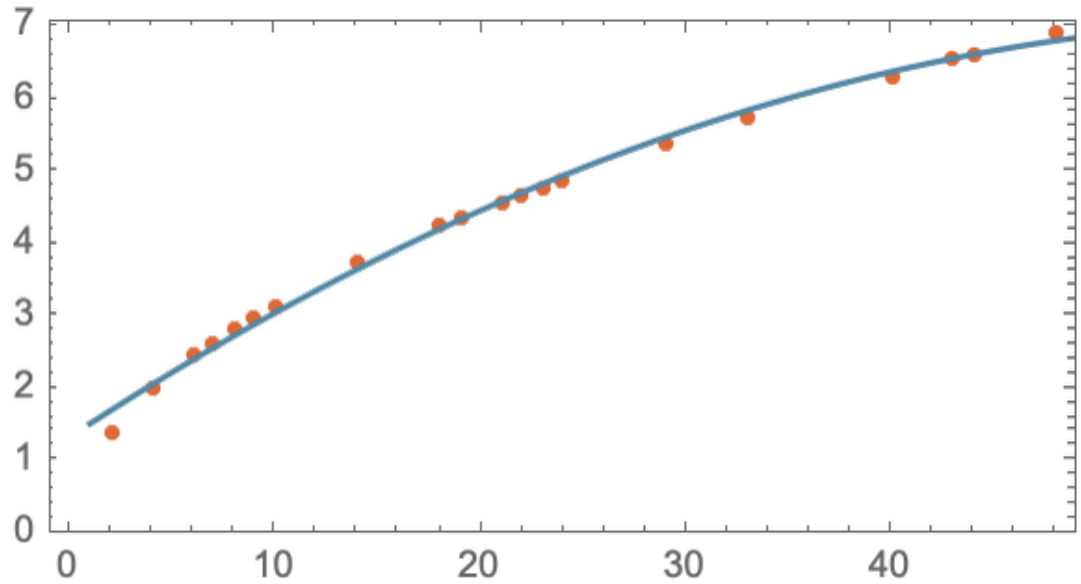
- You could lug that cannon ball up to all the windows and brute strength the problem.



- Or you could attempt to fit a model.
- In this case a linear model. Why? Linear often works and is tractable.

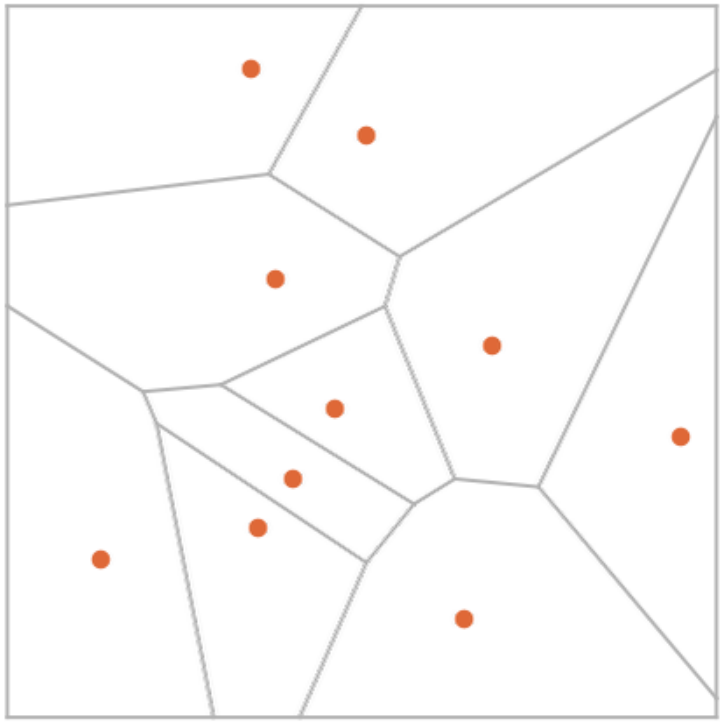


- We can do better in this case with a quadratic.
- $a + bx + cx^2$
- All models have some underlying structure with parameters that can be adjusted. Three above: a, b, and c.
- ChatGPT has 175 billion of them.



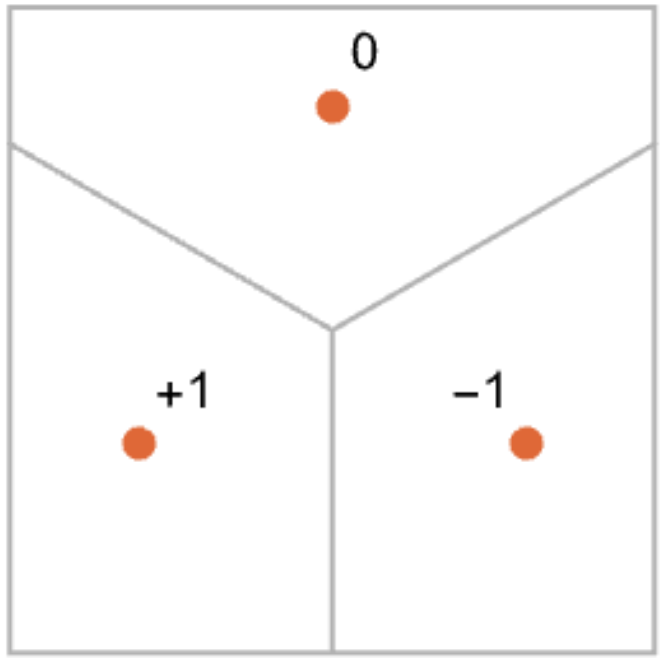
- Imagine we're doing some sort of image recognition task (or trying to find the closest coffee show).

- Voronoi diagram shows the closest shop/answer for any set of coordinates/parameters.

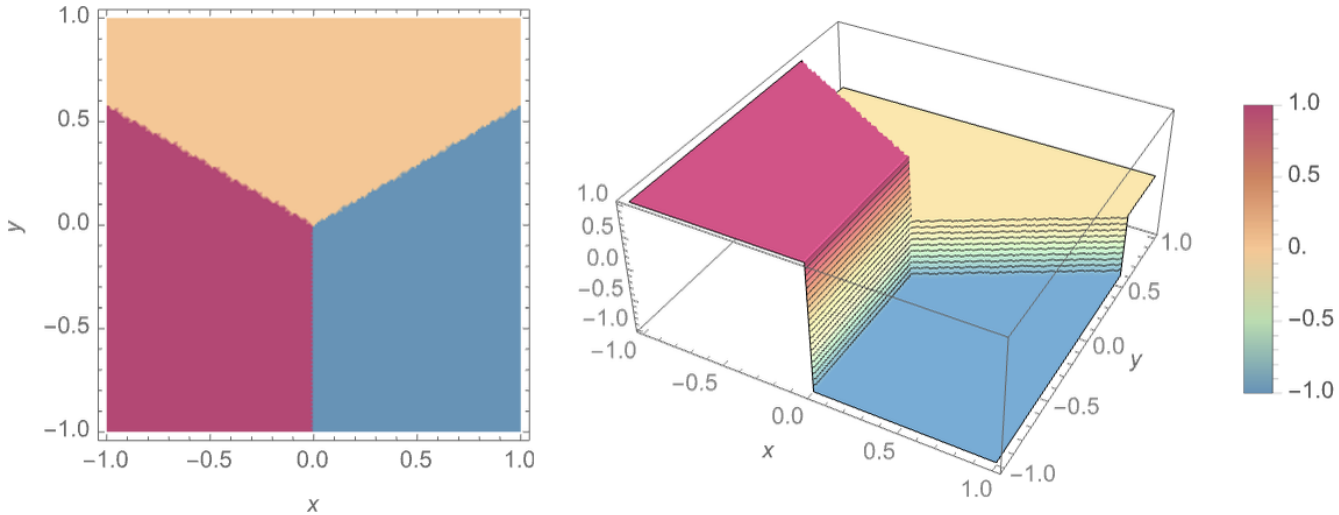


- Let's simplify this.

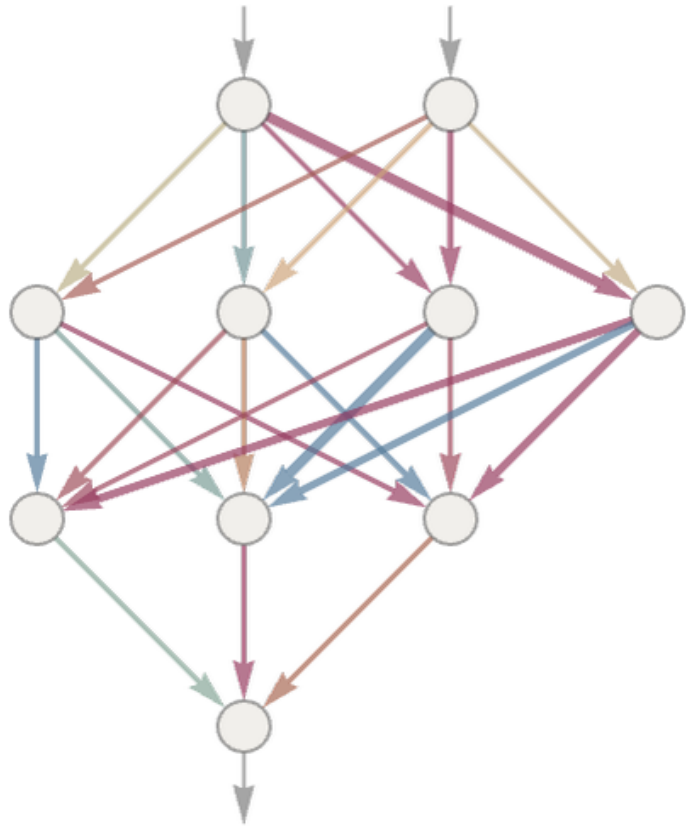
- Enter two variables and ask the model to recognize/classify into the correct region

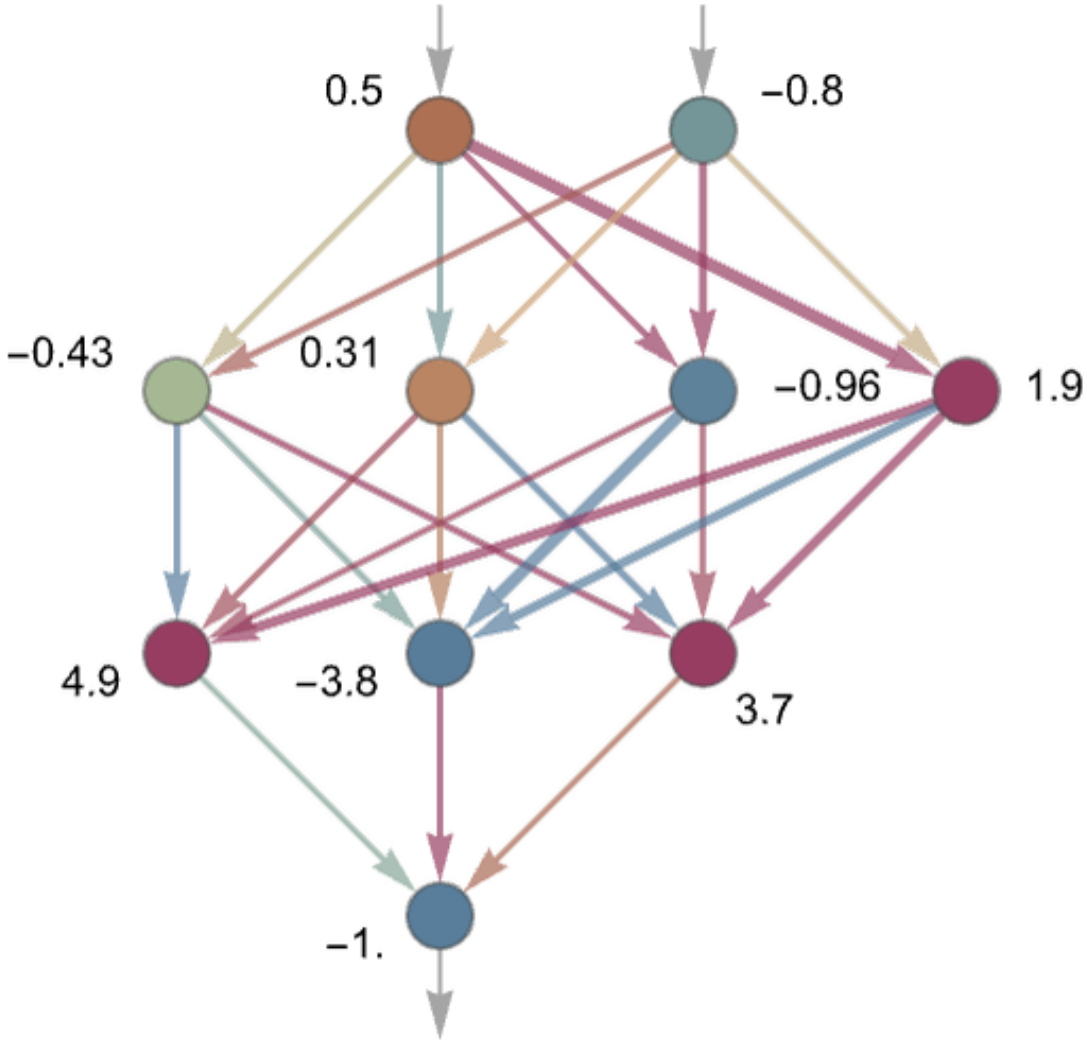


- We want the neural net to figure out a function that looks something like this...

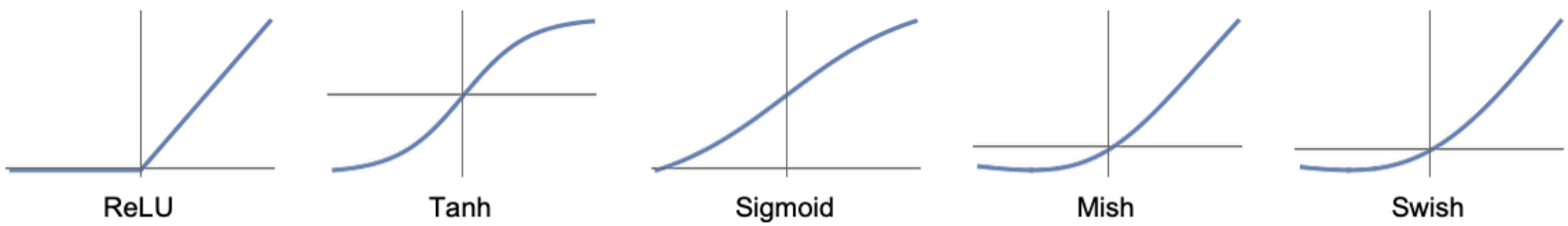


- Here's a simple model of a possible neural net.
- Takes in two inputs in the top two "neurons"
- Each neuron applies some function to the input and then passes the result on to the next layer of neurons.
- Output is the bottom neuron



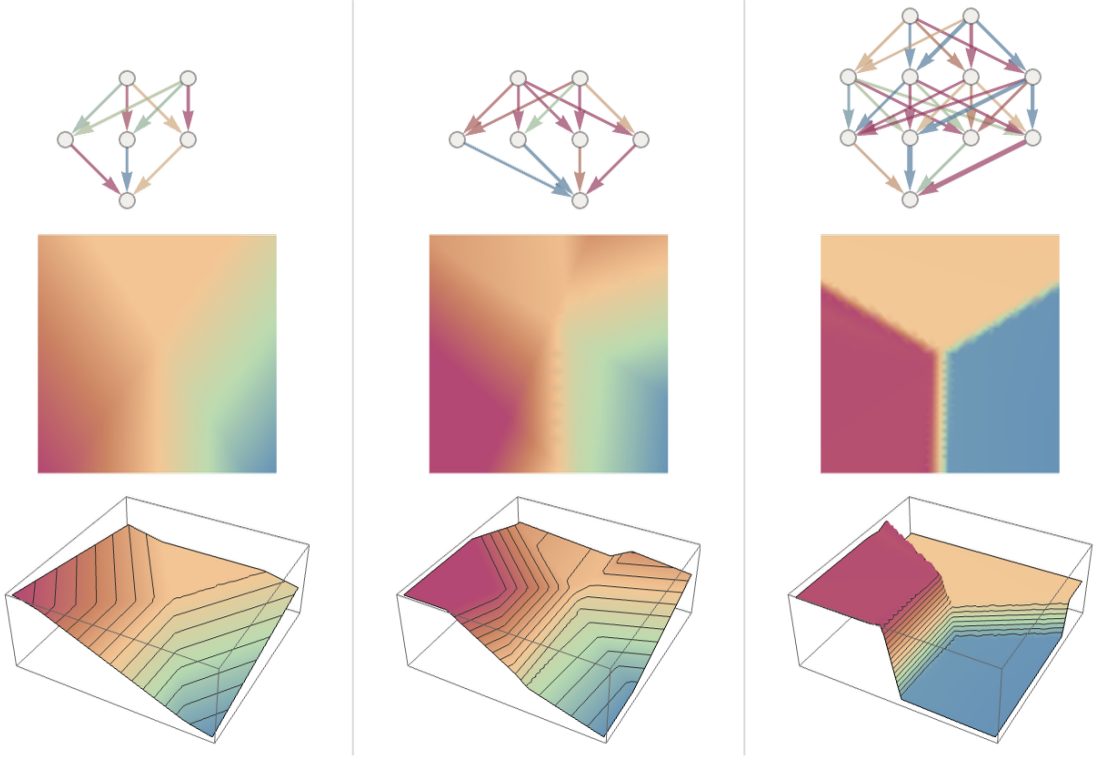

0.5
−0.8
−0.43
0.31
−0.96
1.9
4.9
−3.8
3.7
−1.

- If a neuron has input $x = x_1, x_2...$
- Then the neuron just evaluates $f(w \cdot x + b)$ and passes on to the next layer



- The architecture is chosen and then the model is trained to find the “best” parameters for each neuron.

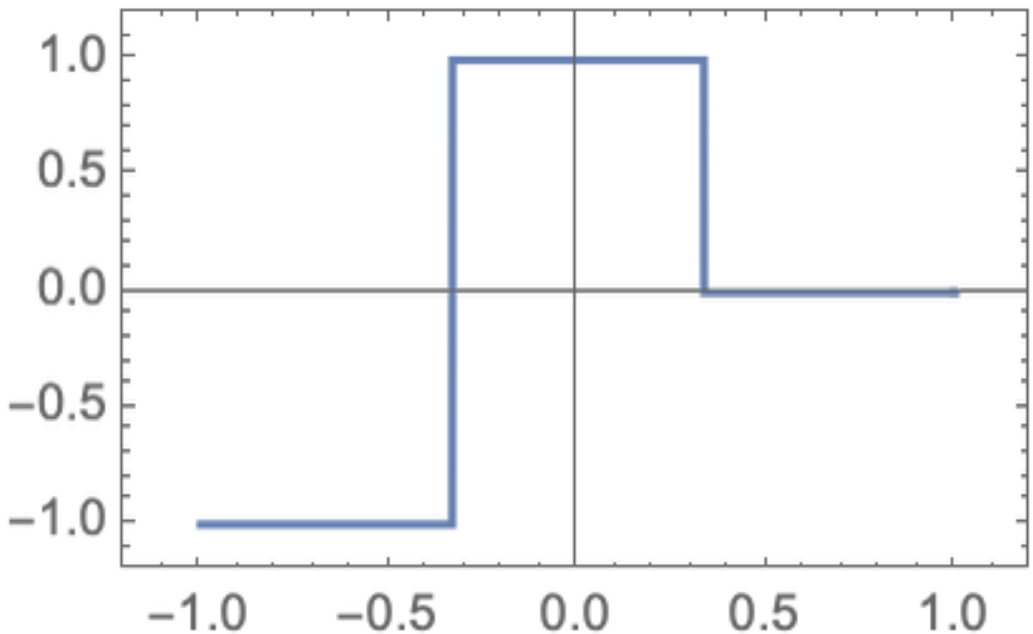
- Here are some examples...



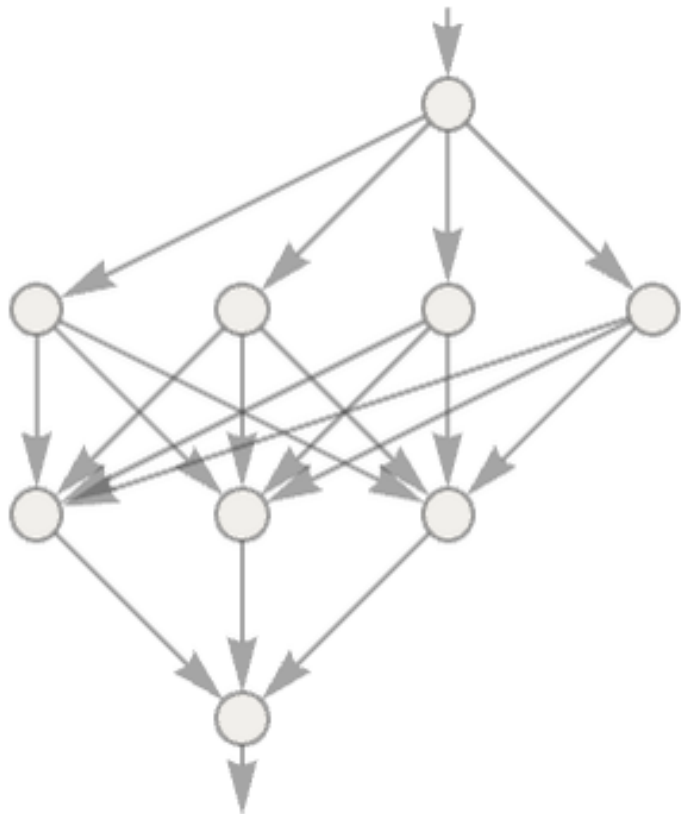
- Do we know "how" the different layers of the net are making the decisions of how to classify?
- Not really.
- How do humans distinguish between a cat and dog?

- Let's think more about training

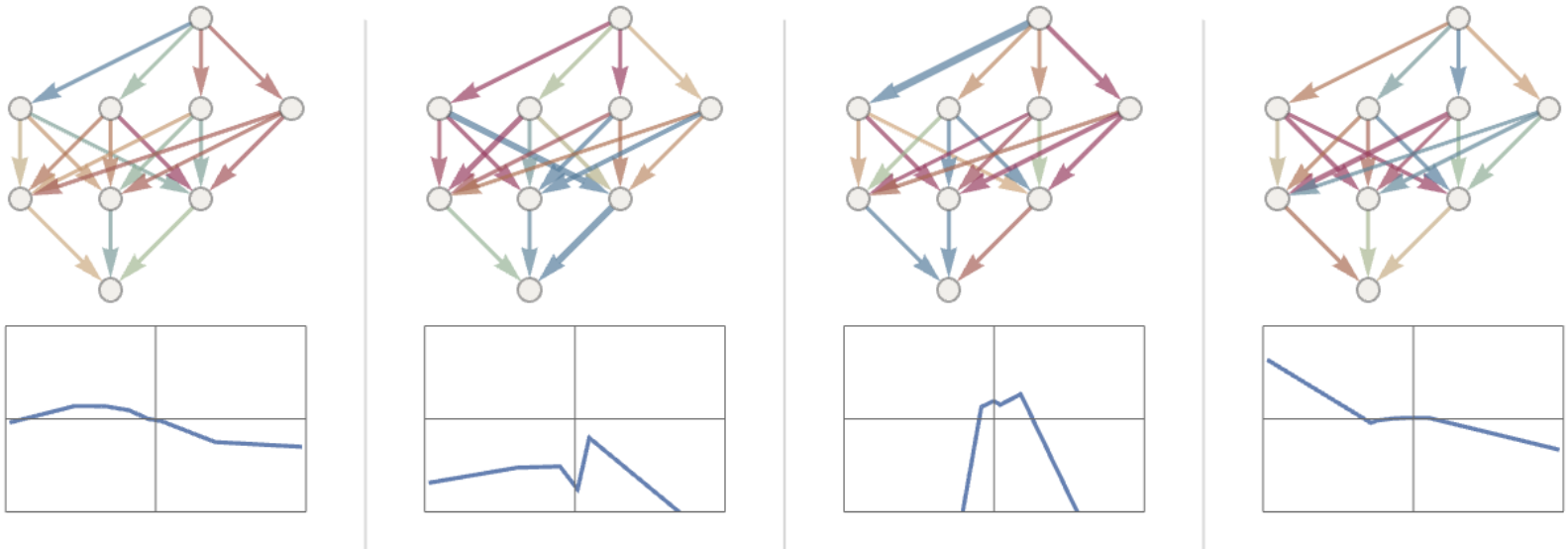
- A simple problem: learn this function.



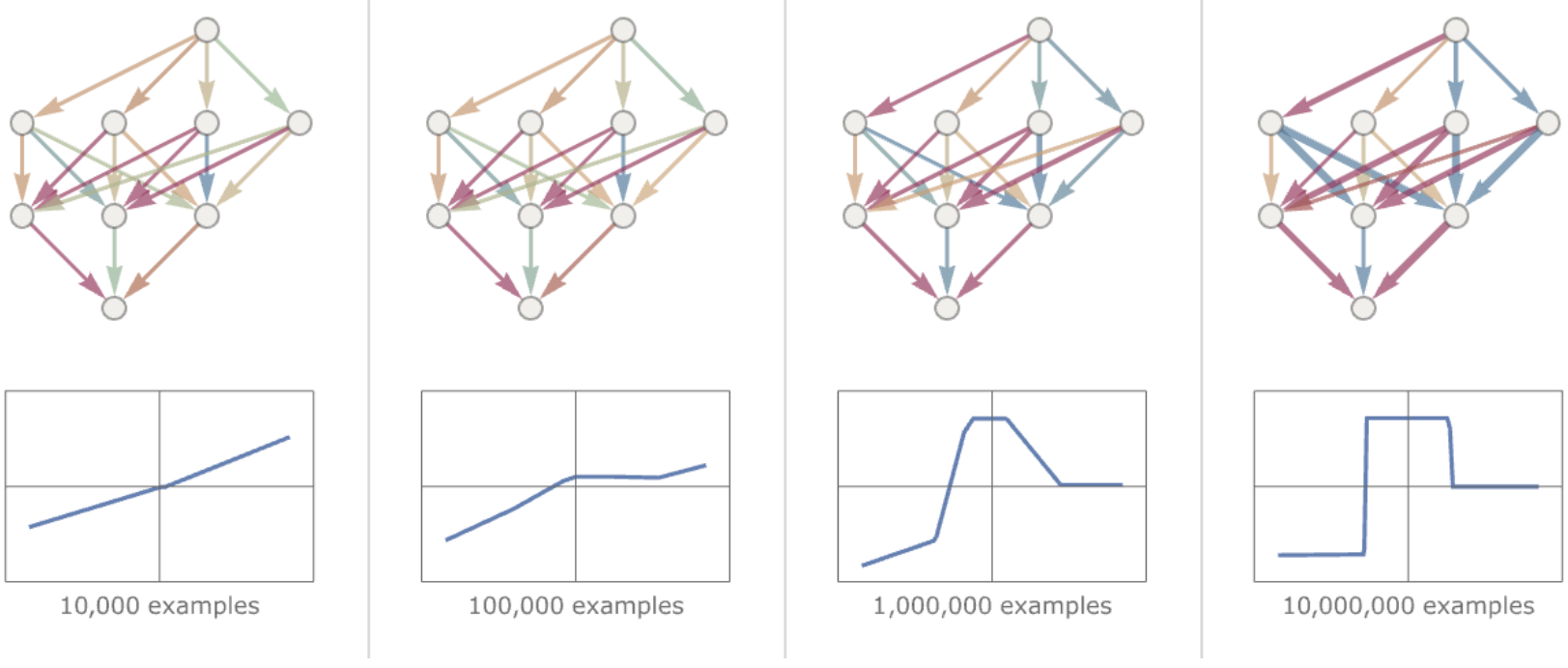
- A simple neural net with one input and on output.



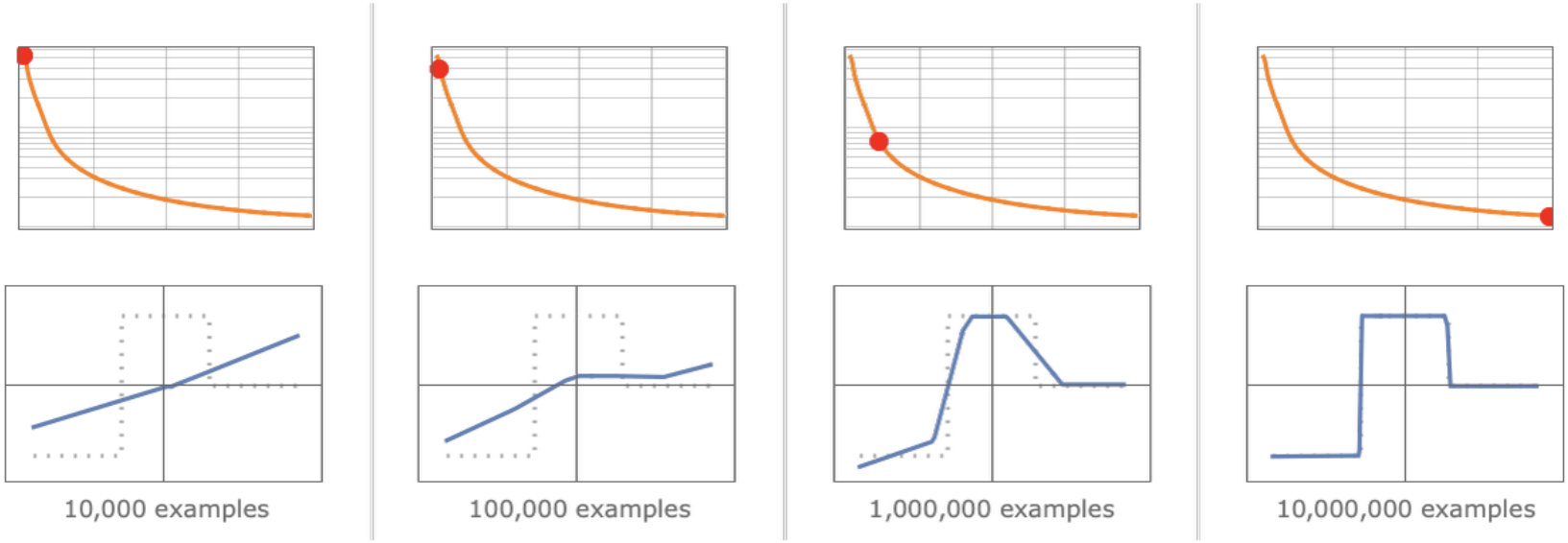
- What is we start with random weights?



- Basic idea is to supply lots of input and output examples to learn from...
- And then to try to find weights that will reproduce these examples

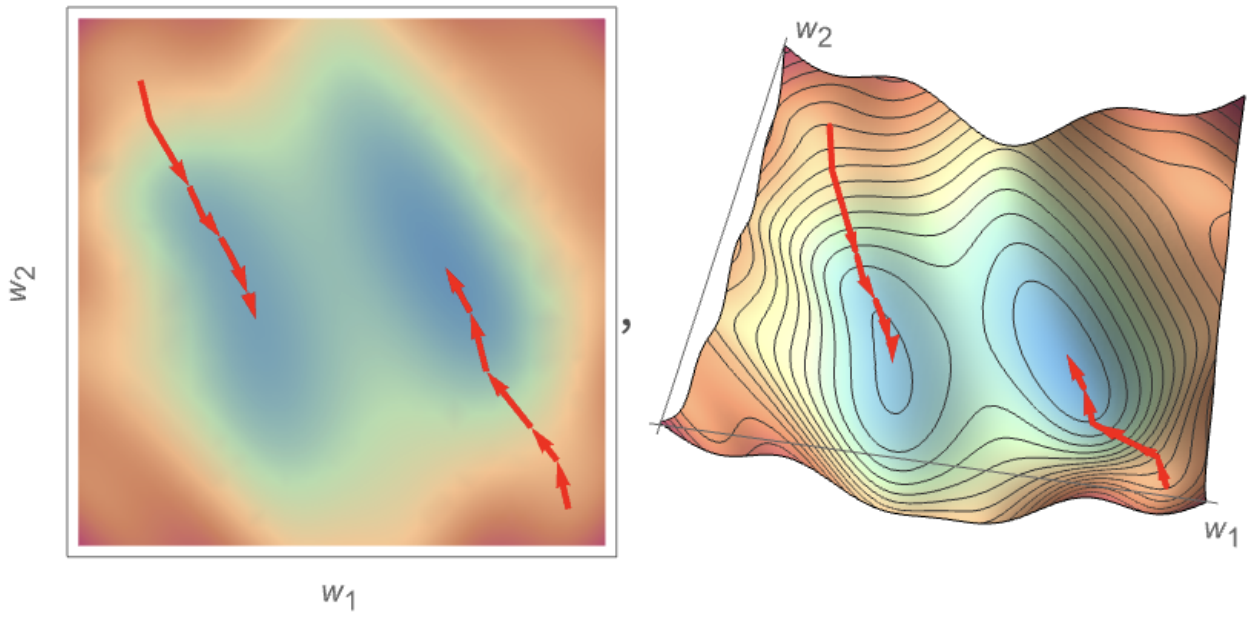


- Just trying to minimize some loss function
- But how do we adjust the weights to move down the loss function?

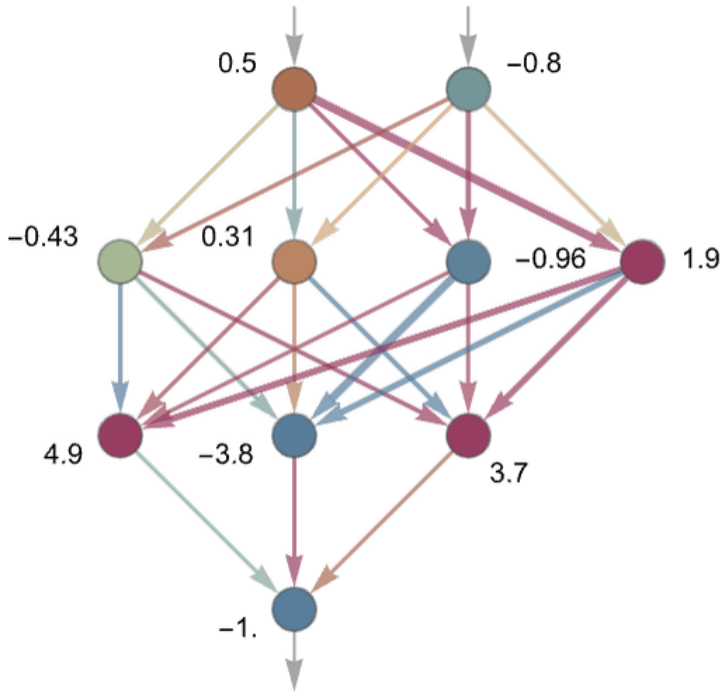


- If we knew the underlying functional form of the loss function, we could use calculus to follow the gradient to the local or global minima

- But we don't.



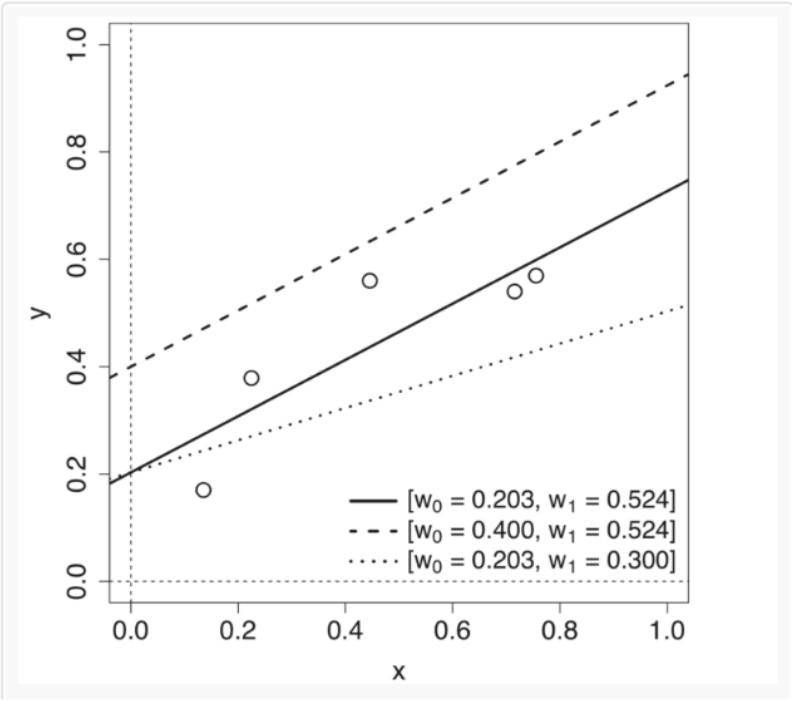
- Recall our simple neural net.

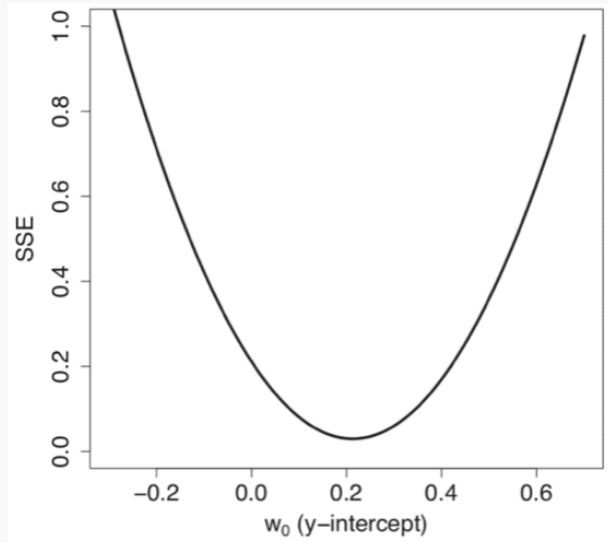


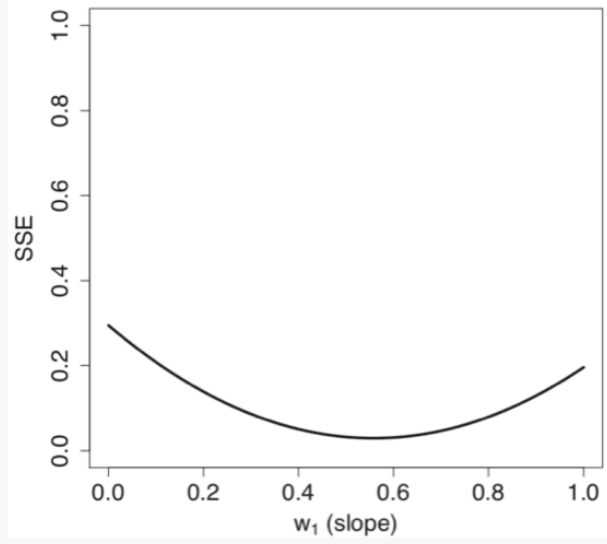
- This can just be written down as an (albeit super ugly) equation.
- We are trying to adjust each parameter so that we are moving "down" the loss function gradient.
- Think of the neural net parameters as variables to optimized entering into the loss function.

$$
\begin{aligned}
&w_{511} f(w_{311} f(b_{11} + x\,w_{111} + y\,w_{112}) + w_{312} f(b_{12} + x\,w_{121} + y\,w_{122}) + \\
&\qquad w_{313} f(b_{13} + x\,w_{131} + y\,w_{132}) + w_{314} f(b_{14} + x\,w_{141} + y\,w_{142}) + b_{31}) + \\
&w_{512} f(w_{321} f(b_{11} + x\,w_{111} + y\,w_{112}) + w_{322} f(b_{12} + x\,w_{121} + y\,w_{122}) + \\
&\qquad w_{323} f(b_{13} + x\,w_{131} + y\,w_{132}) + w_{324} f(b_{14} + x\,w_{141} + y\,w_{142}) + b_{32}) + \\
&w_{513} f(w_{331} f(b_{11} + x\,w_{111} + y\,w_{112}) + w_{332} f(b_{12} + x\,w_{121} + y\,w_{122}) + \\
&\qquad w_{333} f(b_{13} + x\,w_{131} + y\,w_{132}) + w_{334} f(b_{14} + x\,w_{141} + y\,w_{142}) + b_{33}) + b_{51}
\end{aligned}
$$

- Let's consider a simpler example...



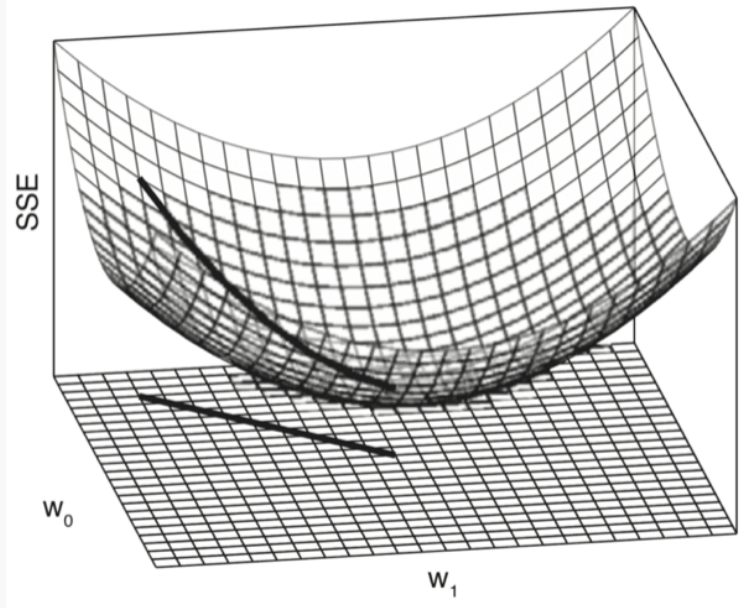




Line Plots of Error (SSE) Profiles Generated When Sweeping Across a Range of Values for the Slope and Intercept

- We don't know the equation for the loss function though, so how do we figure out the derivative of it with respect to some parameter/variable at a given point to know "which way" do go?

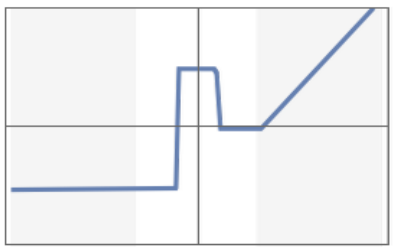
- Taylor Series, the Chain Rule, and Back Propagation



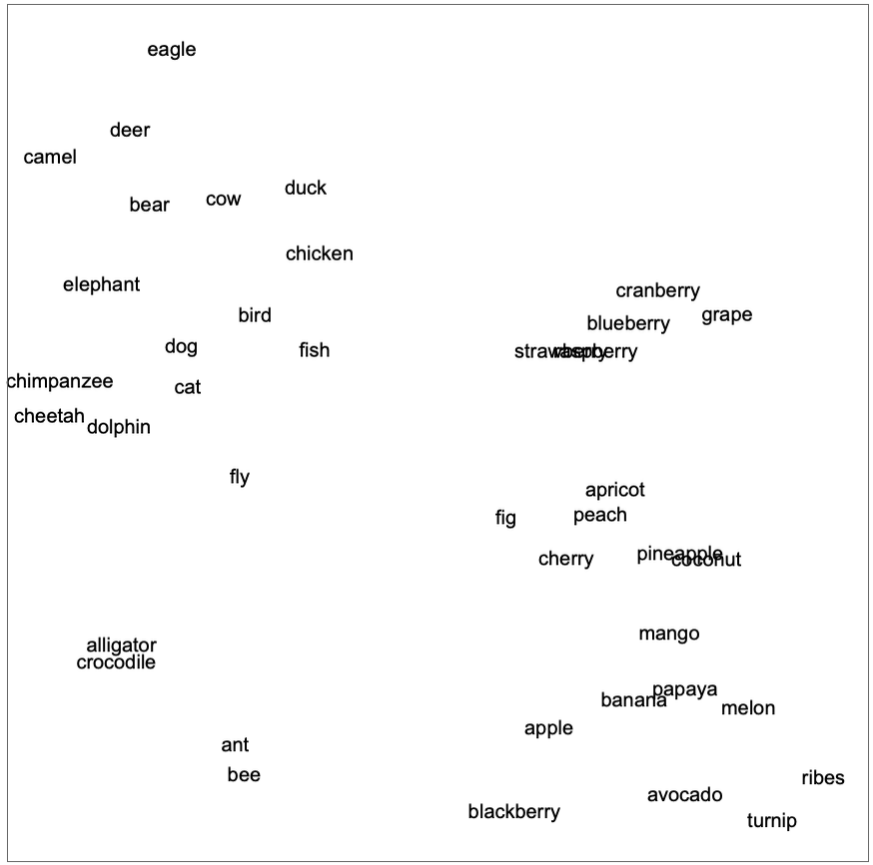
Three-Dimensional Plot of the Error (SSE) Surface Generated When Both Slope and Intercept are Varied

- Back to our simple problem.

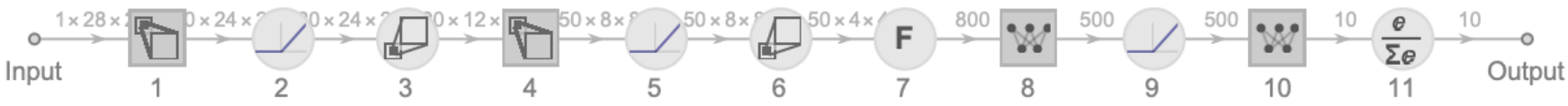
- There are many possible solutions, but they will lead to different extrapolations.




Back to Embeddings



We can get a word embedding from looking at the penultimate layer in the neural net



Here's the last layer in a neural net trying to recognize a handwritten 4

$$\{1.42071\times10^{-22},\ 7.69857\times10^{-14},\ 1.9653\times10^{-16},\ 5.55229\times10^{-21},\ 1.,$$
$$8.33841\times10^{-14},\ 6.89742\times10^{-17},\ 6.52282\times10^{-19},\ 6.51465\times10^{-12},\ 1.97509\times10^{-14}\}$$

Here's the penultimate layer

$$\{-26.134,\ -6.02347,\ -11.994,\ -22.4684,$$
$$24.1717,\ -5.94363,\ -13.0411,\ -17.7021,\ -1.58528,\ -7.38389\}$$