Neural Networks

INF1-CG Week 4

Maithilee Kunda

- Supervised learning
- Multi-layer neural networks
- Gradient descent and backpropagation

Supervised Learning

- A branch of machine learning (different from unsupervised learning, reinforcement learning, etc etc.)
- Also one type of learning that humans do, kindof (maybe)
- **Supervised Learning:**
  Construct a decision boundary using a set of data points where you know the inputs and the true output
  - This is also called “classification” - dividing data points into distinct classes
  - Can also have “regression” - learn a continuous function, not just a boundary

- Supervised learning always has two parts:
  - A way to represent a decision boundary of some shape – “hypothesis space”
  - A way to define/adjust a specific boundary to fit some data points – “learning”

Examples of Supervised Learning Methods:

A way to represent a DB + how to adjust the DB based on data

- A **perceptron** is a linear decision boundary
  - Perceptron learning rule is a way to set weights to match a set of data
- A **multi-layer perceptron (MLP)** is a type of neural network that can represent a non-linear decision boundary
  - Need a slightly different learning rule to set weights
- A **neural network** is a more general term that includes MLPs and also networks with fancier neurons
  - Backpropagation is one of the most common learning rules for NNs
- A **decision tree** is a different kind of decision boundary altogether! Not a neural network at all. Represents complex logical functions.
  - There is a decision tree learning rule called ID3 that takes a bunch of data points and constructs a decision tree from it

What is the goal of supervised learning?

- But wait… if we already know the inputs and true output for our data, why bother with all this decision boundary business?

- **Generalization: Learn from one set of data (training data), and be able to apply the same decision boundary to make decisions about new data points never seen before**

- Make decision = apply label = classify = predict

- **Overfitting** – learning a decision boundary that overly matches the training data but does not generalize well

Examples of supervised learning

- Look at pictures of venomous snakes to learn which ones to avoid in the wild
  - Training data: Photos of individual snakes
  - Generalization: New individual snakes never seen, but same species
- Learn to recognize letters of the alphabet
  - Training data: Particular fonts, handwriting, illustrations
  - Generalization: New fonts, handwriting, never before seen
- Predict what consumers will buy from your shopping website
  - Training data: Past consumers, how they browsed, what they bought
  - Generalization: New website visitors
- Decide if an MRI scan shows evidence of a tumor or not
  - Training data: Scans from a bunch of patients and their diagnoses
  - Generalization: Scans from new patients
- Many uses, both scientific and commercial/application-oriented

What are “Features” in supervised learning?

- The inputs!
- Often, we have to decide what these are

- UCI Machine Learning repository is a famous place to find supervised learning datasets on all kinds of topics
  - https://archive.ics.uci.edu/
  - Truly one of the classic, foundational resources that drove a huge amount of ML research for the past few decades

Artificial Neural Networks

- Current “champion” among machine learning methods for supervised learning
- But other methods are still used! Depends on the problem

- Inspired by how neurons in the brain integrate information and pass it along to other neurons
- But… many many manymany differences between ANNs and real brain neurons and networks

- **Don’t fall for the hype! ANNs do NOT mean that a machine learns or thinks “just like humans do”**
- **ANN is just a complex nonlinear function, and “learning” is just setting weights to define a particular function**

The Guardian

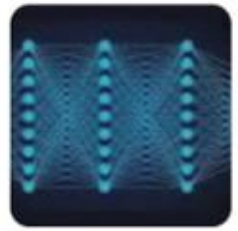
Google's DeepMind makes AI program that can learn like a human | Science

Program brings artificial general intelligence a step c
to solve fresh problems.

14 Mar 2017

AI Business

How Neural Networks Can Think Like H ns And Why It Matters



Some n etworks are to human cogniti ever before,
acc

Technology Networks

Nanowire Neural Network Le s Just L

Artificial neural networks could provide a sc to many
medical gnosis, face identification system
30 Nov 202

eu cience News

Inspired AI Learns Lik mans

researchers designed a ew del inspired by the human brain's efficiency.
T l allows AI neuron to re e f dback and adjust in real
20 J 20

ighe Di nsion—Li to
th n Brain and Achiev Int

Resea how modeling the human brai
beyond it t lim into human-like cognition.
15 Jul 2025

Qua

Some

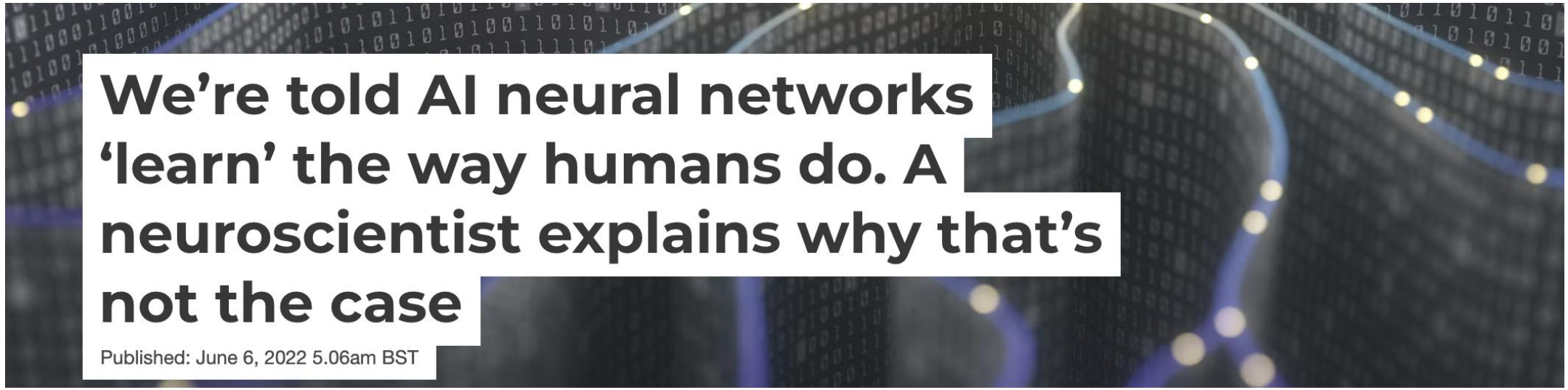
The researchers — led by per Beguš (opens a new tab), a c
the University of California, B eley — compared the brain...
22 May 2023

The Independent

Artificial intelligence that mimics the brain needs sleep just like humans, study reveals

Researchers at Los Alamos discover neural network reaps benefits 'equivalent to a good night's rest'

8 Jun 2020



We’re told AI neural networks ‘learn’ the way humans do. A neuroscientist explains why that’s not the case

Published: June 6, 2022 5.06am BST

Shutterstock

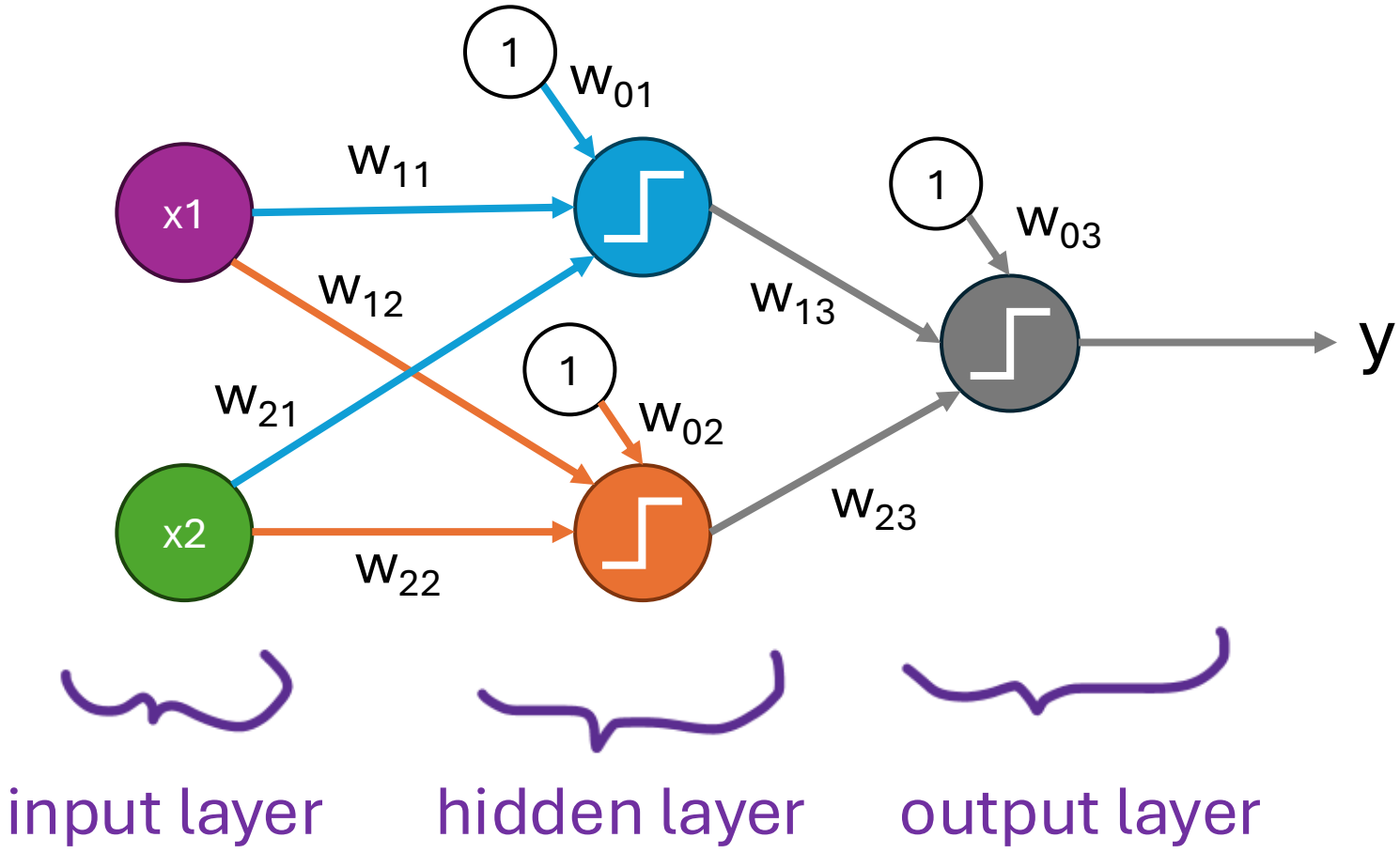
Recently developed artificial intelligence (AI) models are capable of many impressive feats, including recognising images and producing human-like language. But just because AI can perform human-like behaviours doesn’t mean it can think or understand like humans.

Author

James Fodor
PhD Candidate in Cognitive Neuroscience, The University of Melbourne

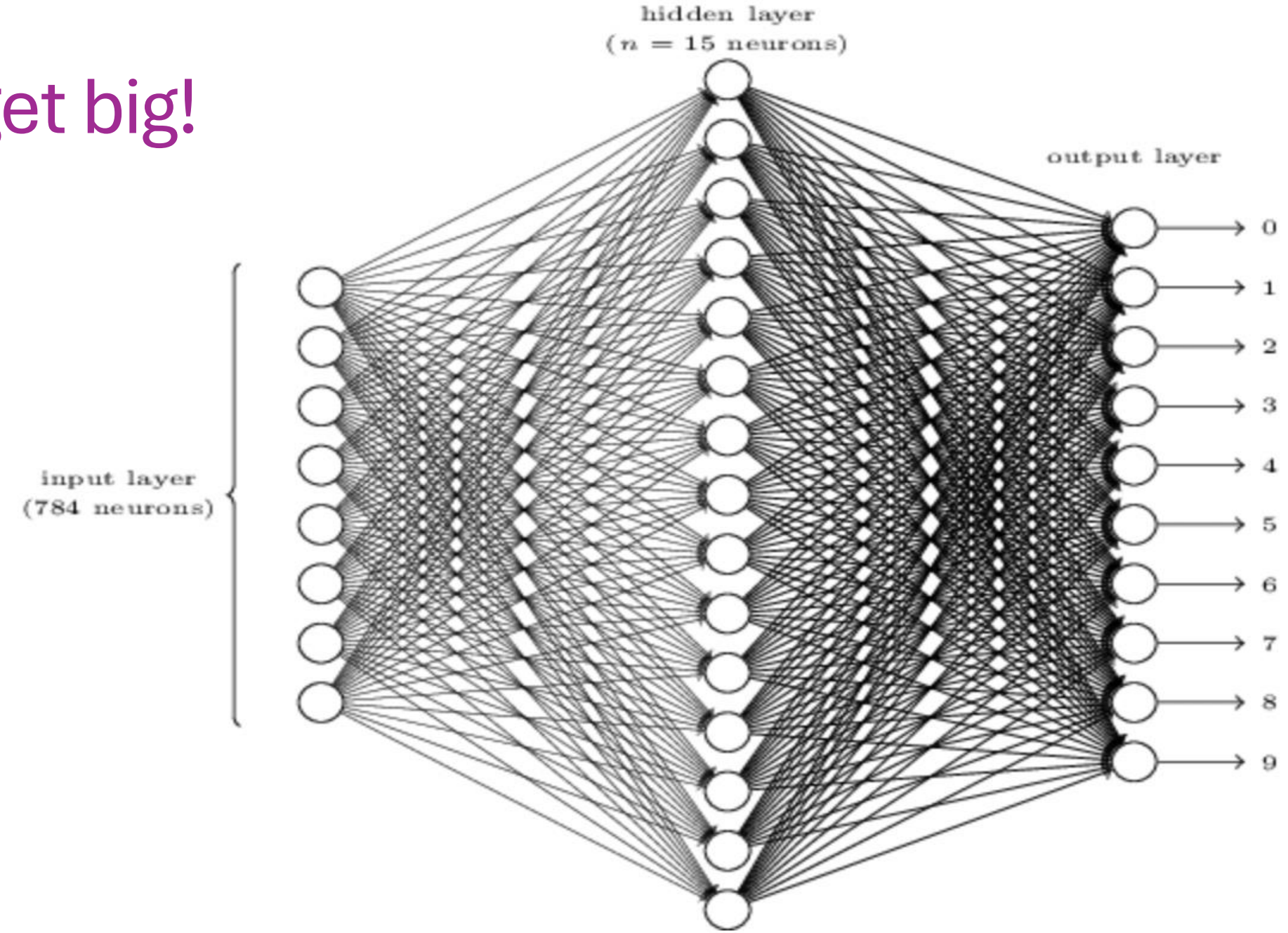
- https://theconversation.com/were-told-ai-neural-networks-learn-the-way-humans-do-a-neuroscientist-explains-why-thats-not-the-case-183993

Multi-layer networks



- Nodes in the hidden layer can represent values that are based on, but different from, the inputs
- This network has
  - two input nodes,
  - one hidden layer with two nodes,
  - one output node
- Hidden layers allow you to build more complex functions (i.e. more complex decision boundaries)

Can get big!


hidden layer
(n = 15 neurons)
output layer
input layer
(784 neurons)
0
1
2
3
4
5
6
7
8
9

Can have fancy shapes… this is “AlexNet” (2012)

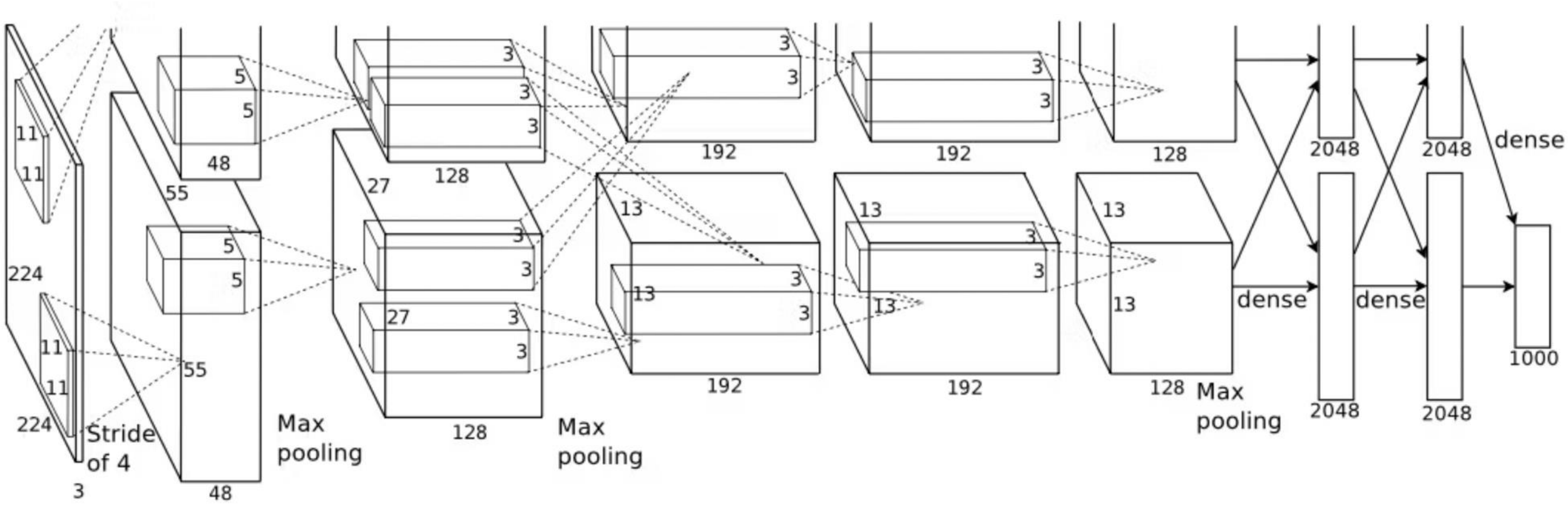


Figure 2: An illustration of the architecture of our CNN, explicitly showing the delineation of responsibilities between the two GPUs. One GPU runs the layer-parts at the top of the figure while the other runs the layer-parts at the bottom. The GPUs communicate only at certain layers. The network’s input is 150,528-dimensional, and the number of neurons in the network’s remaining layers is given by 253,440–186,624–64,896–64,896–43,264–4096–4096–1000.

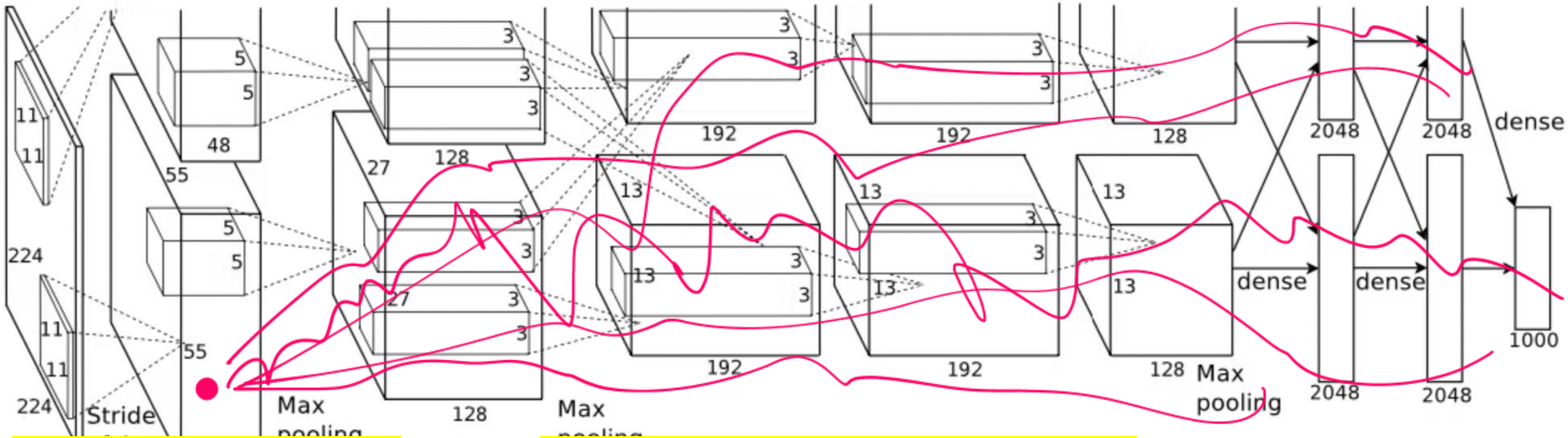
Very big and complex networks

- Are good because they can represent very complex decision boundaries

- Are bad because it is difficult to find a set of weights that will make the network do what you want!

- Every weight influences not just its own little node, but ALSO every other node that is somewhere downstream in the network

Can have fancy shapes… this is “AlexNet” (2012)



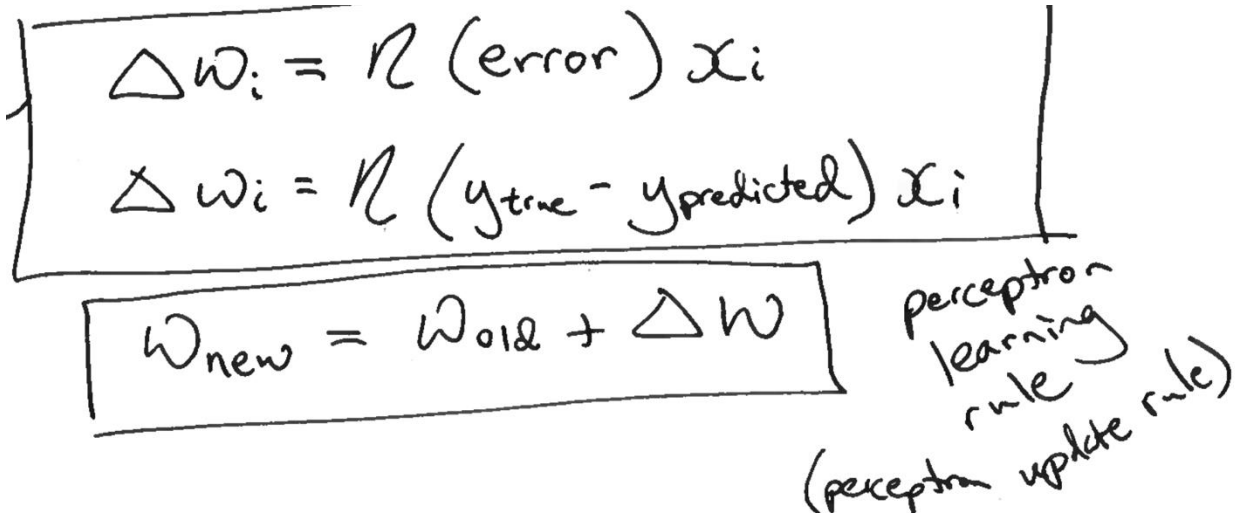
Change a weight here…

It will impact LOTS and LOTS of other calculations throughout the network!

…the archit… …delineation of responsibilities between the two GPUs. One GPU runs… …le the other runs the layer-parts at the bottom. The GPUs communicate only at certain layers. The network’s input is 150,528-dimensional, and the number of neurons in the network’s remaining layers is given by 253,440–186,624–64,896–64,896–43,264–4096–4096–1000.

So how to pick a good set of weights?

- Perceptron update rule: Iteratively tweak weights to reduce the error

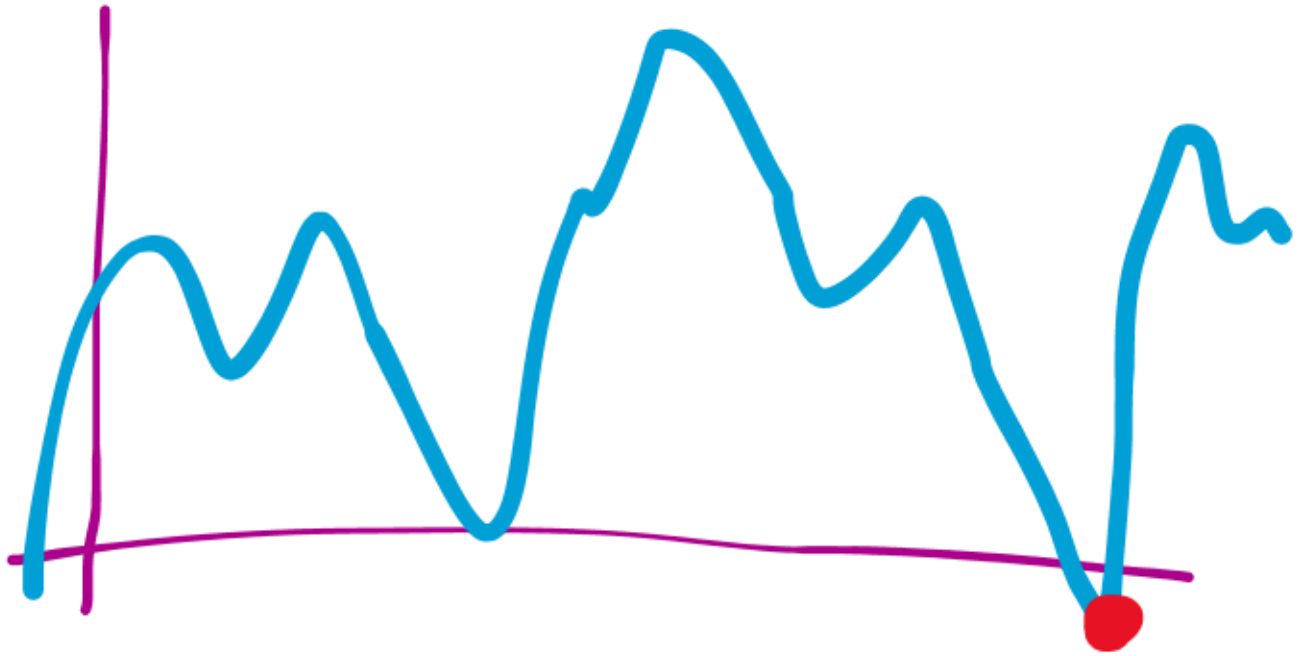


- **Fancier idea: Iteratively tweak weights to follow the <u>slope</u> of the error function**

Gradient descent is a type of optimization

- Gradient descent is a general optimization algorithm
- **Optimization: Given an arbitrary function over some inputs, we want to find which inputs minimize (or maximize) the value of that function**

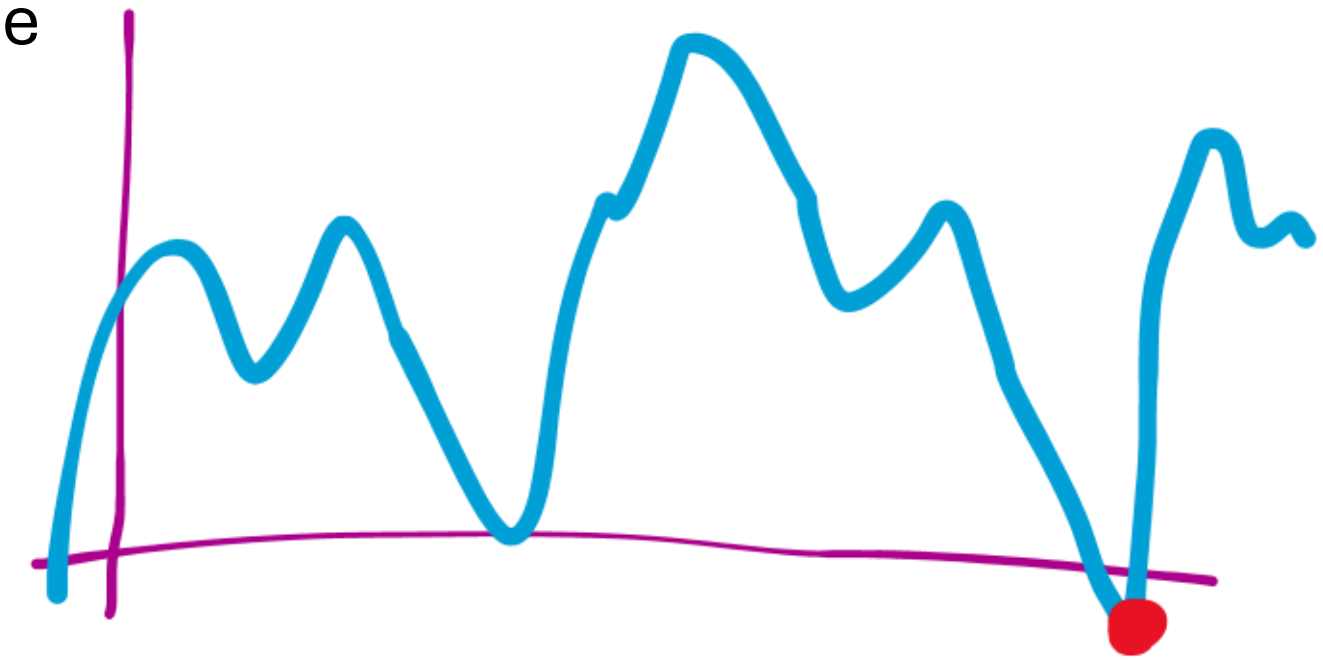
- Exhaustive search would be easy and effective!
- But what if you can't?
- **Hill climbing: Start at a random point, look at your neighbors, and go downhill**
- Often works with enough starts



More clever math….

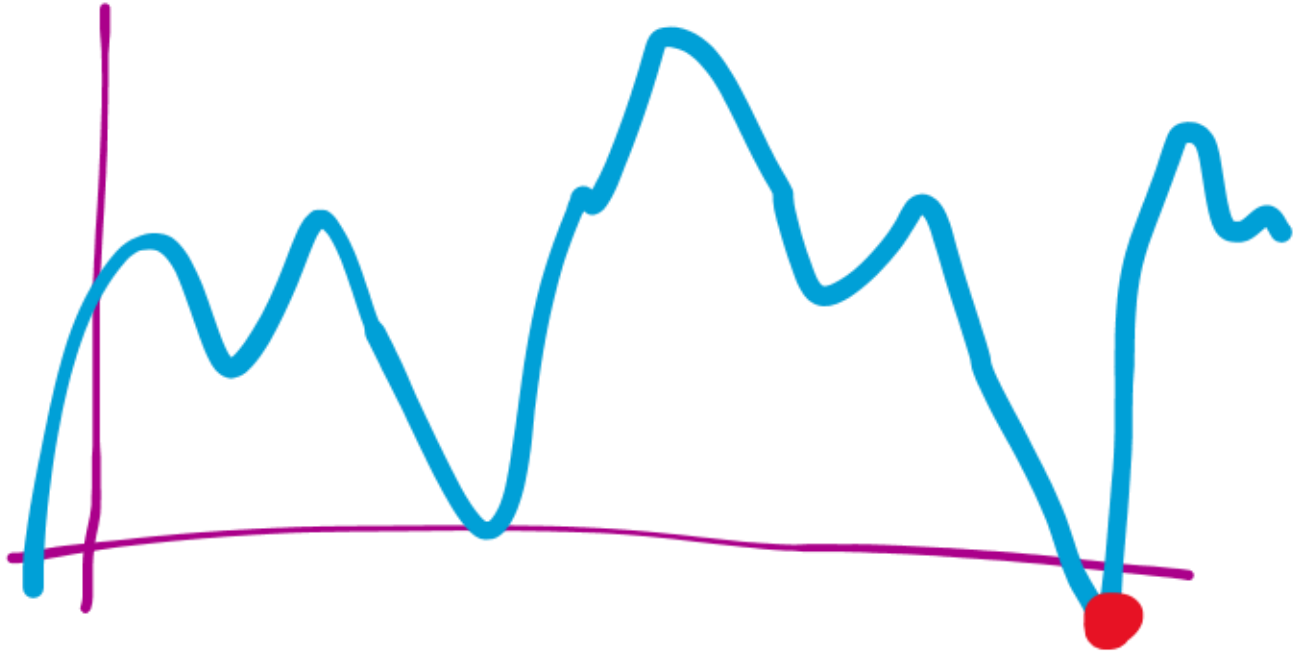
- Instead of looking at your neighbors, discretely, why not just look at the slope of the spot right under your feet?

- How can we calculate the slope of a function?
- Take the derivative!
- I.e., the **gradient** (for a multivariable function)



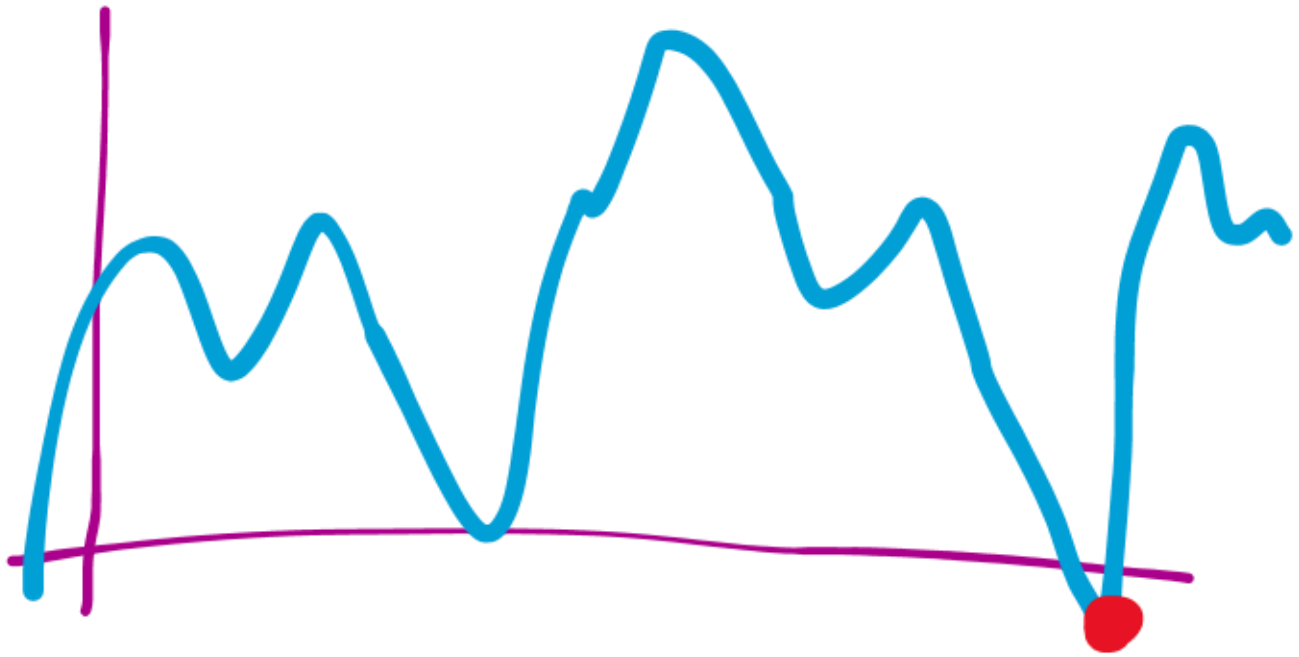
Gradient descent

- Function you want to minimize: **F**
- Calculate the gradient: **∇F**
- Wherever you are, just look at the value of **∇F** at that point…
- Take a step in the direction **-∇F**
  - Minus because we want to minimize the function F
- How big of a step?
  - Learning rate **η**
- Step = **η * -∇F**



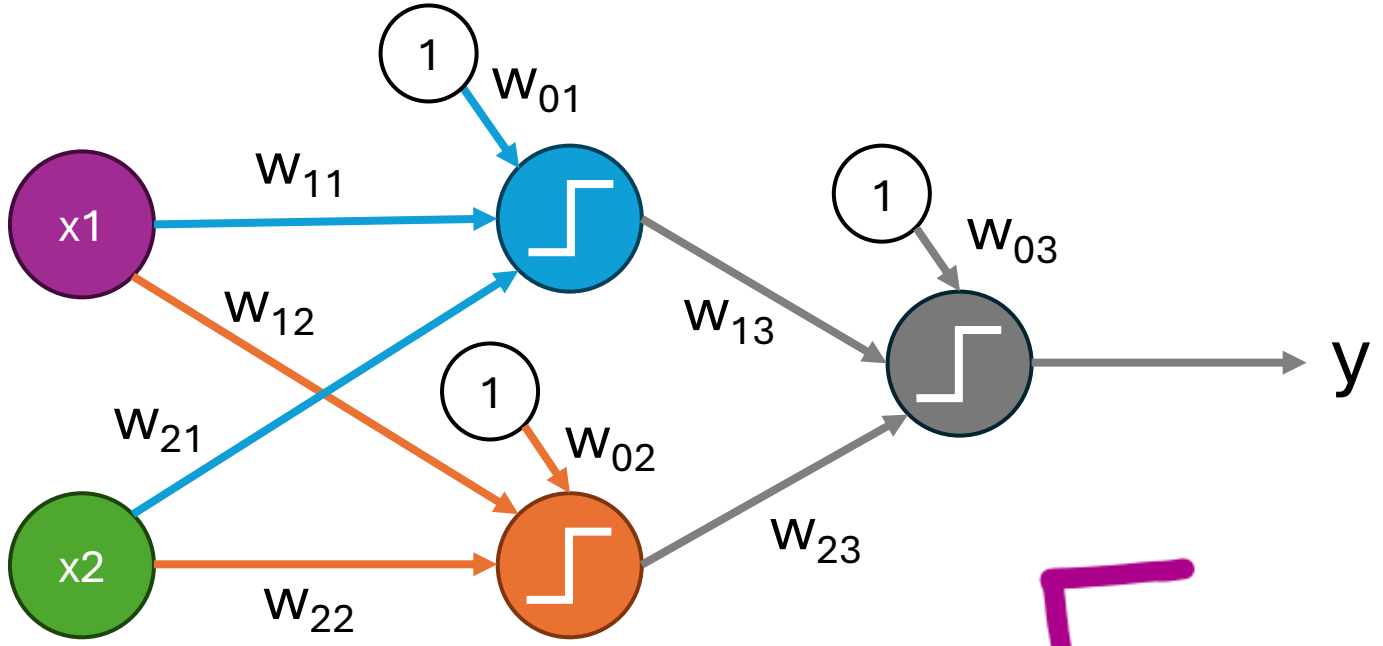
Gradient descent for neural networks

- Think about the **error** of a neural network
- A function defined over….
- Not over the inputs to the network! **Defined over the space of all possible weights**
- If the error function is differentiable… then we can use gradient descent to try to adjust the weights in order to minimize the error!



Problem…

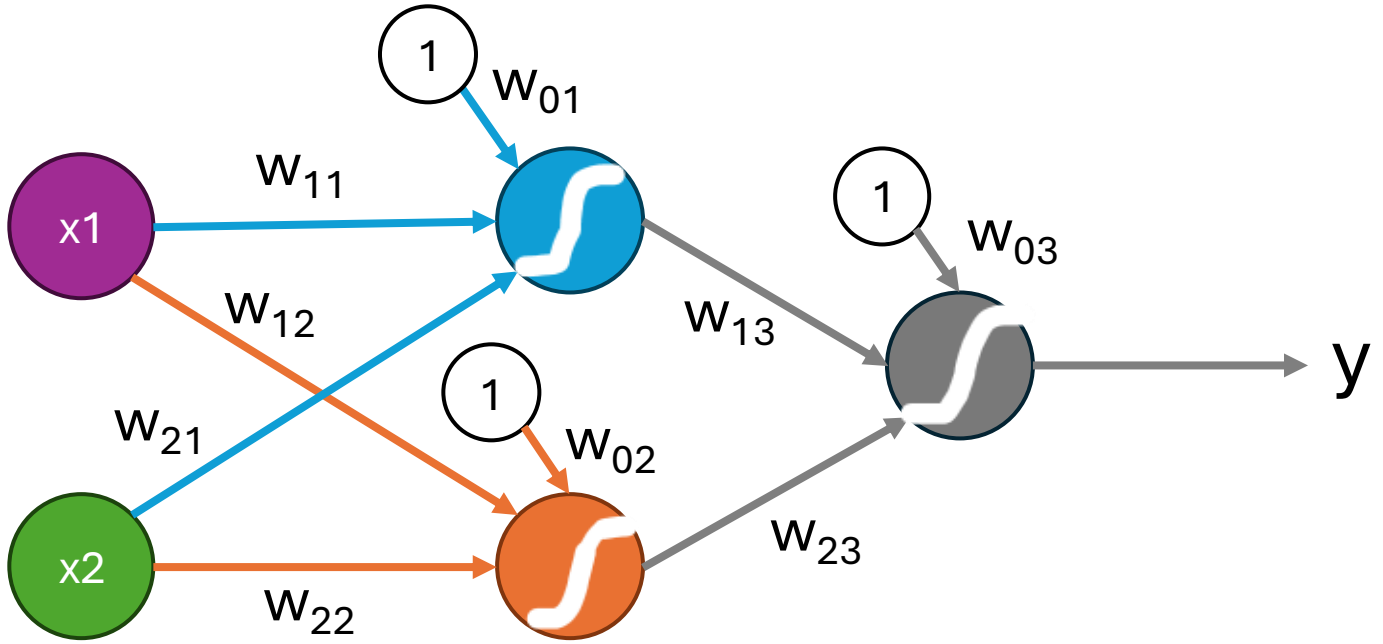
- If the error function includes the step function, it is not differentiable!



- Solution: Smooth out the step function! "Sigmoid" function is a common one

Gradient descent in a sigmoid neural network

- This whole thing is differentiable!
- And we can define the error so that it is easily differentiable too



- “Backpropagation” is a method for doing gradient descent where you repeatedly compute the derivative of the error function with respect to the weights, starting at the output layer and working your way ”backwards” through the network

How much data to use to calculate error at each iteration?

- True gradient descent: Use all the data at every time step
- Get the most accurate idea of which way the slope is pointing, given ALL of our data
- Tough to compute, if we have a lot of data

- Stochastic (or “batch”) gradient descent: Use a random batch of the data at every time step
- Epoch – one pass through all of the training data
- Usually train over multiple epochs

Play with a NN!

- https://playground.tensorflow.org