Introduction to Natural Language Processing

BUS 243 F: Spring 2023

Yeabin Moon

Lecture 5

Questions in homework 2

- Some discussions…
- Homework2

Hmm. So where are we?

- Text classification and topic modeling may seem like all we need for natural language processing
- You can handle most natural language problems
  - Detail might matter
- Mathematically represent language units - frequency

Individual Frequency based approach

- The BOW representation is inherently high dimensional
  - Usually possible to find a linear classifier that perfectly fits the training data
  - even to fit any arbitrary labeling of the training instances
  - In general, it really works well
- Word frequencies are meaningful in isolation
  - Can offer independent evidence about the instance label
- NLP has historically focused on linear classification

Now what?

- But in recent years, nonlinear classifiers have swept through NLP, and are now the default approach for many tasks (Manning, 2015)
- You can see many NLP textbooks dedicated to “Deep learning”
- There are at least three reasons for this change

Technological progress: method

- There have been rapid advances in **deep learning**
    - a family of nonlinear methods that learn complex functions of the input through multiple layers of computation
    - Goodfellow et al., 2016

Technological progress: resource

- While CPU speeds have plateaued, there have been rapid advances in specialized hardware called graphics processing units (GPUs) thanks to gaming industry
  - which have become faster, cheaper, and easier to program
- Many deep learning models can be implemented efficiently on GPUs, offering substantial performance improvements over CPU-based computing

Text representation breakthrough

- Deep learning facilitates the incorporation of **word embeddings**
  - Dense vector representations of words
  - Dense vector?
- Word embeddings can be learned from large amounts of unlabeled data, and enable generalization to words that do not appear in the annotated training data
  - Kind of unsupervised learning
- Let’s learn Distributed representations

How do we represent the meaning of a word?

- Definition: meaning
  - What is meant by a word, text, concept, or action
- Commonest linguistic way of thinking of meaning:
  - Signifier (symbol) ←→ signified (idea or thing)
    - Denotational semantics
  - What’s the problem?

How do we have usable meaning in a machine?

- Previously commonest NLP solution: Use, e.g. WordNet, a thesaurus containing lists of **synonym sets** and **hypernyms**

e.g., synonym sets containing "good":

```
from nltk.corpus import wordnet as wn
poses = { 'n':'noun', 'v':'verb', 's':'adj (s)', 'a':'adj', 'r':'adv'}
for synset in wn.synsets("good"):
    print("{}: {}".format(poses[synset.pos()],
          ", ".join([l.name() for l in synset.lemmas()])))
```

```
noun: good
noun: good, goodness
noun: good, goodness
noun: commodity, trade_good, good
adj: good
adj (sat): full, good
adj: good
adj (sat): estimable, good, honorable, respectable
adj (sat): beneficial, good
adj (sat): good
adj (sat): good, just, upright
…
adverb: well, good
adverb: thoroughly, soundly, good
```

e.g., hypernyms of "panda":

```
from nltk.corpus import wordnet as wn
panda = wn.synset("panda.n.01")
hyper = lambda s: s.hypernyms()
list(panda.closure(hyper))
```

```
[Synset('procyonid.n.01'),
Synset('carnivore.n.01'),
Synset('placental.n.01'),
Synset('mammal.n.01'),
Synset('vertebrate.n.01'),
Synset('chordate.n.01'),
Synset('animal.n.01'),
Synset('organism.n.01'),
Synset('living_thing.n.01'),
Synset('whole.n.02'),
Synset('object.n.01'),
Synset('physical_entity.n.01'),
Synset('entity.n.01')]
```

Problems with resources like WordNet

- A useful resource but missing nuance:
  - *Proficient* is listed as a synonym for *good*
    - Only correct in some contexts
- Missing new meanings of words
- Subjective
- Requires human labor to create and adapt
- Can’t be used to accurately compute word similarity

Representing words as discrete symbols

- In traditional NLP, we regard words as discrete symbols
  - Such symbols for words can be represented by one-hot vectors
  - Hotel, conference, motel
    - motel: [0 0 0 0 1 0 0 0]
    - hotel: [0 0 1 0 0 0 0 0]
    - a ***localist*** representation
- Hard to represent synonym
- Vector dimension issue
- Challenging in vector reasoning

Problem with words as discrete symbols

- E.g: in web search, if a user searches for “**Seattle motel**”, we’d like to match documents containing “**Seattle hotel**”
  - But see: [0 0 0 0 1 0 0 0] , [0 0 1 0 0 0 0 0]
  - Theses two vectors are orthogonal
    - No natural notion of similarity for one-hot vectors
- Could try to rely on WordNet’s list of synonyms to get similarity?
  - But it is well-known to fail badly

Distributed Representations

- To overcome these limitations, methods to learn low-dimensional representations were devised
  - use neural network architectures to create dense, low-dimensional representations of words and texts
- The meaning of a word can be understood from the context in which the word appears
  - Connotation: meaning is defined by context
  - Opposed to denotation: the literal meaning of any word

Representing words by their context

- **Distributional semantics**: A word’ meaning is given by the words that frequently appear close-by
  - J. R. Firth 1957: You shall know a word by the company it keeps
  - One of the most successful ideas of modern statistical NLP
- When a word *w* appears in a text, its **context** is the set of words that appear nearby
  - Within a fixed-size window

Context, context, context

- We use the many contexts of *w* to build up a representation of *w*
- The following context words will represent banking

…*government debt problems turning into* **banking** *crises as happened in 2009*…

…*saying that Europe needs unified* **banking** *regulation to replace the hodgepodge*…

…*India has just given its* **banking** *system a shot in the arm*…

Word vectors (embeddings)

- We will build a dense vector for each word, chosen so that it is similar to vectors of words that appear in similar contexts, measuring similarity as the vector dot (scalar) product

$$\textit{banking} = \begin{pmatrix} 0.286 \\ 0.792 \\ -0.177 \\ -0.107 \\ 0.109 \\ -0.542 \\ 0.349 \\ 0.271 \end{pmatrix} \qquad \textit{monetary} = \begin{pmatrix} 0.413 \\ 0.582 \\ -0.007 \\ 0.247 \\ 0.216 \\ -0.718 \\ 0.147 \\ 0.051 \end{pmatrix}$$

- Note: **word vectors** are also called (**word**) **embeddings** or (**neural**) **word representations**
- They are a **distributed** representation

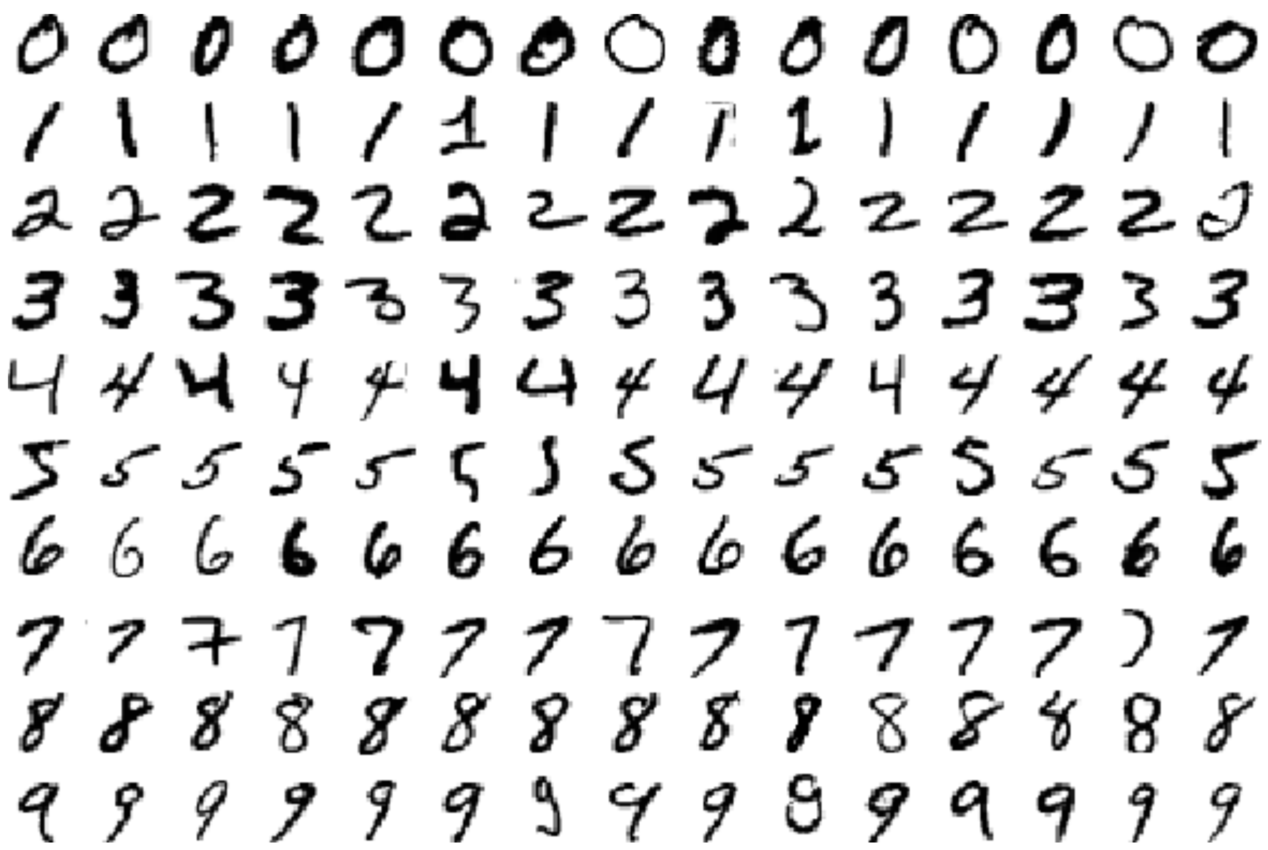
What is Neural Network?

- This is an extremely active field of artificial computer intelligence often referred to as deep learning
- Super popular these days, but many fail to understand this correctly
- What is your understanding?
    - What kind of problems can this solve in particular?
- Linear regression vs. neural net?
    - What is a linear regression by the way?

Limits of Traditional Computer Programs

- Why exactly are certain problems so difficult for computers to solve?
- Machines are good
  - Performing arithmetic really fast
  - Following explicitly a list of instructions
- Suppose need to do some heavy financial number crunching
  - You cannot beat the machine at all!

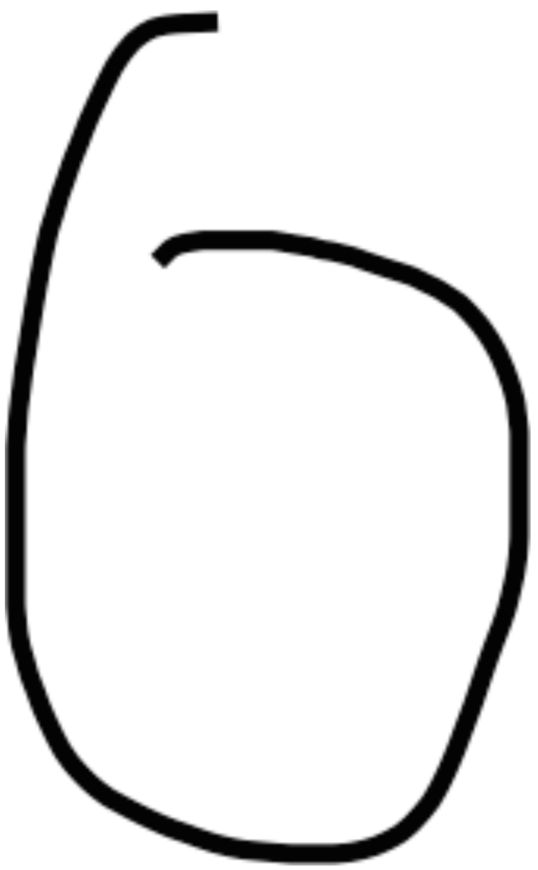
Can we beat the machine for image recognition?



How to recognize zero?

- Suppose we want to know how to recognize 0 image
- What rules could we use to tell one digit from another?
- We might state that we have a zero if our image has only a single, closed loop
  - Is this sufficient condition?

How about this: zero or six



Don’t know how it’s done by our brains

- Establish some cutoff between the starting and ending point of the loop?
- How about threes and fives? Fours and nines?
- We can add more and more rules, or ***features***, through careful observation and months of trial and error
- But it’s quite clear that this isn’t going to be an easy process
- Wait, how did we learn?
  - Did you ever learn those rules?
  - Did you ever learn…?

Mechanics of Machine Learning

- A lot of the things we learn in school growing up have much in common with traditional computer programs
  - learn how to multiply numbers, solve equations, and take derivatives by internalizing a set of instructions
- But the things we learn at an extremely early age, the things we find most natural, are learned by example, not by formula
  - How did you recognize a zero, dog, …?

Learning process

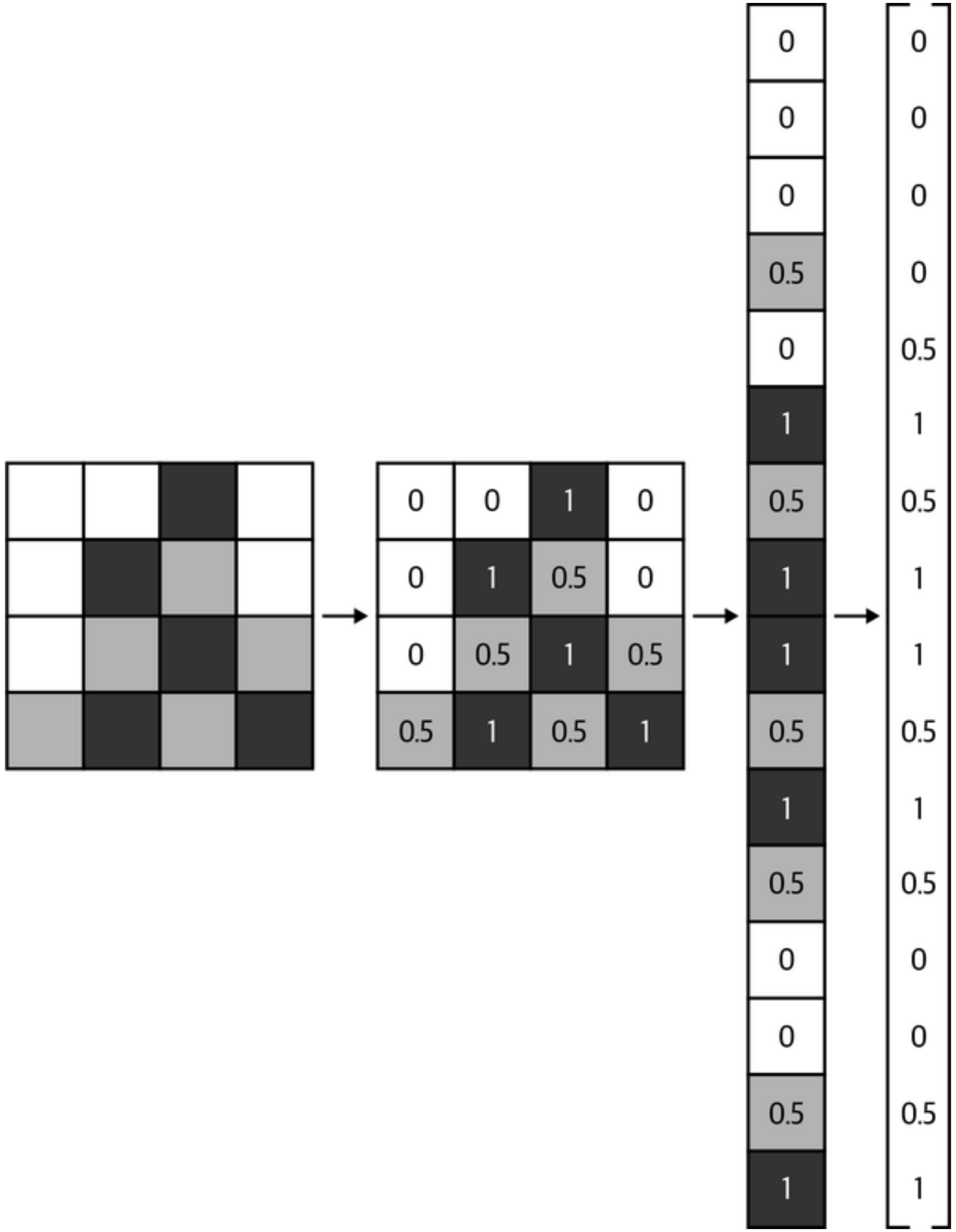
- When we were born, our brains provided us with a model that described how we would be able to see the world
  - I am just saying…
- As we grew up, that model would take in our sensory inputs and make a guess about what we were experiencing.
  - If that guess was confirmed by our parents, our model would be reinforced.
  - If our parents said we were wrong, we'd modify our model to incorporate this new information.
- Over our lifetime, our model becomes more and more accurate as we assimilate more and more examples

continue

- Deep learning is a subset of a more general field of AI called machine learning, which is predicated on this idea of learning from example
- In machine learning, instead of having rules
  - we give it a model with which it can evaluate examples,
  - and a small set of instructions to modify the model when it makes a mistake
- We expect that, over time, a well-suited model would be able to solve the problem accurately

Little math

- Let's define our model as $h(\boldsymbol{x}, \theta)$
  - $\boldsymbol{x}$: input vector
    - If it were a grayscale image, express it like:
  - The input θ is a vector of the params
- Machine learning program tries to perfect the values of these params as it is exposed to more and more examples



Quick example: predict exam performance

- $\boldsymbol{x} = [x_1 \;\; x_2]^T$ where x_1 is hours of sleep and x_2 is hours of studying
- Want to know whether we perform above or below the class average
- Our goal might be to learn a model $h(\boldsymbol{x}, \theta)$ with $\theta = [\theta_0, \theta_1, \theta_2]^T$

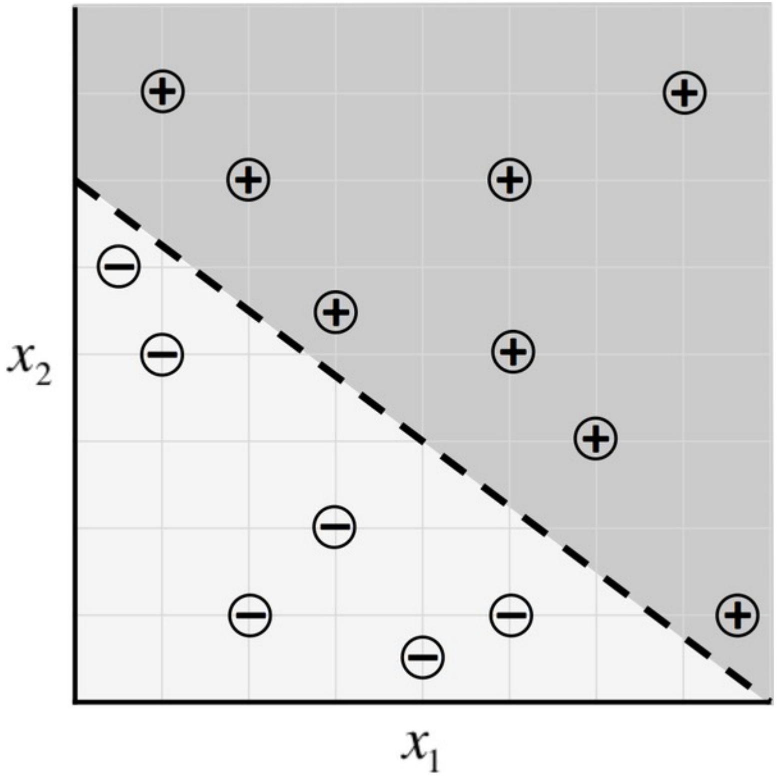
$$h(\mathbf{x}, \theta) = \begin{cases} -1 & \text{if } \mathbf{x}^T \cdot \begin{bmatrix} \theta_1 \\ \theta_2 \end{bmatrix} + \theta_0 < 0 \\ 1 & \text{if } \mathbf{x}^T \cdot \begin{bmatrix} \theta_1 \\ \theta_2 \end{bmatrix} + \theta_0 \geq 0 \end{cases}$$

continue

$$h(\mathbf{x}, \theta) = \begin{cases} -1 & \text{if } \mathbf{x}^T \cdot \begin{bmatrix} \theta_1 \\ \theta_2 \end{bmatrix} + \theta_0 < 0 \\ 1 & \text{if } \mathbf{x}^T \cdot \begin{bmatrix} \theta_1 \\ \theta_2 \end{bmatrix} + \theta_0 \geq 0 \end{cases}$$

- Want to learn θ such that the model makes the right predictions
- This is called a linear *perceptron*

Sample data for our exam predictor algorithm and a potential classifier



Suppose the following θ the model makes the correct prediction on every data point:

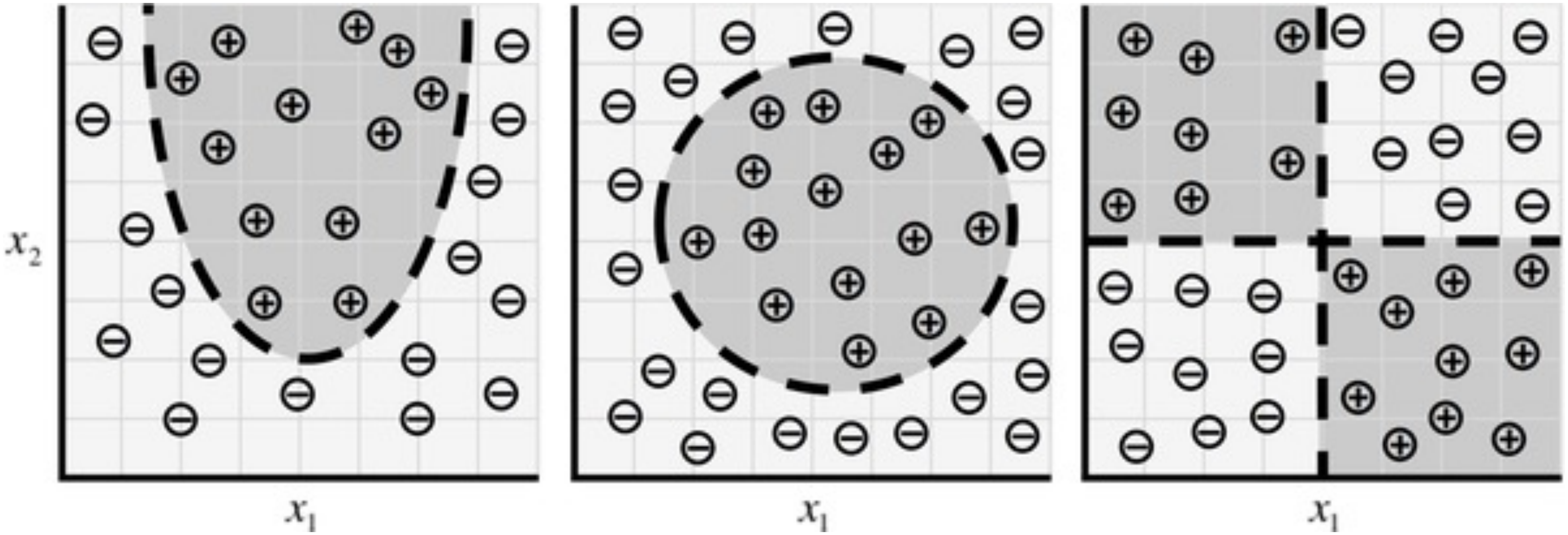
$$h(\mathbf{x}, \theta) = \begin{cases} -1 & \text{if } 3x_1 + 4x_2 - 24 < 0 \\ 1 & \text{if } 3x_1 + 4x_2 - 24 \geq 0 \end{cases}$$

Note that this model classifies perfectly

Limitations?

- How do we come up with an optimal value for the parameter vector in the first place?
  - Called **optimization**
  - An optimizer aims to maximize the performance of a machine learning model by iteratively tweaking its parameters until the error is minimized
    - Gradient descent, SGD, …
- What if the data looks ugly?

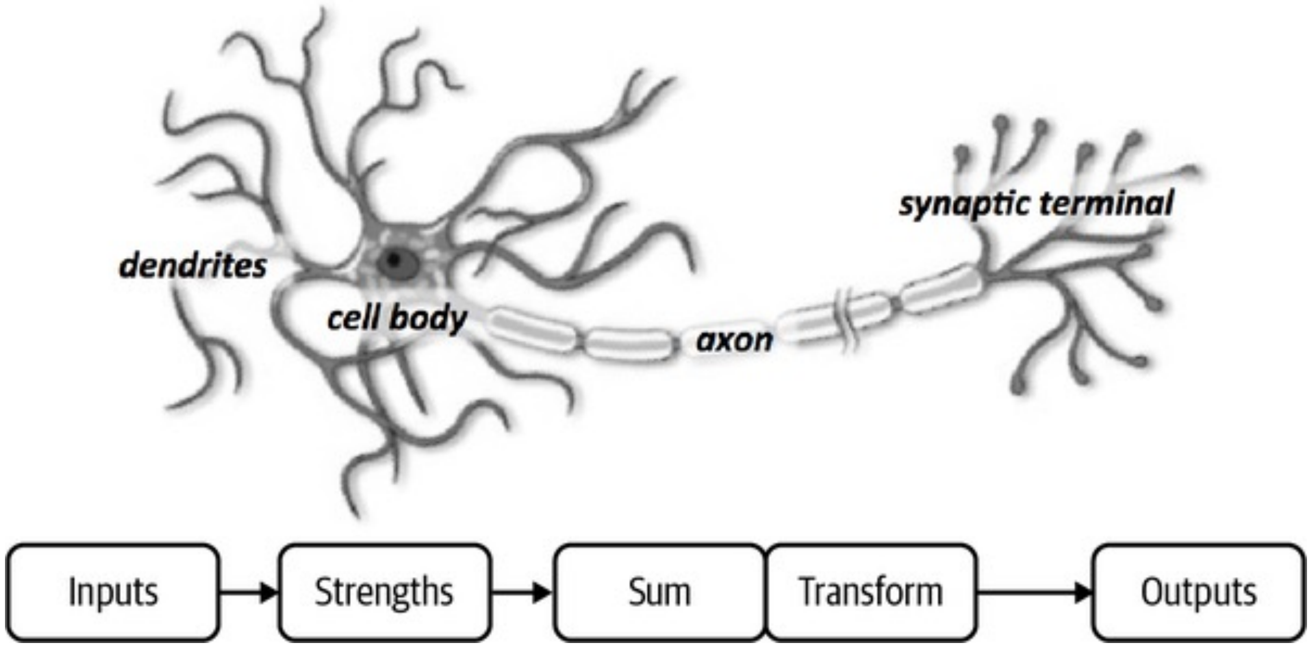
Linear is just a linear



- To accommodate this complexity, try to build models that resemble the structures utilized by our brains – deep learning

The Neuron

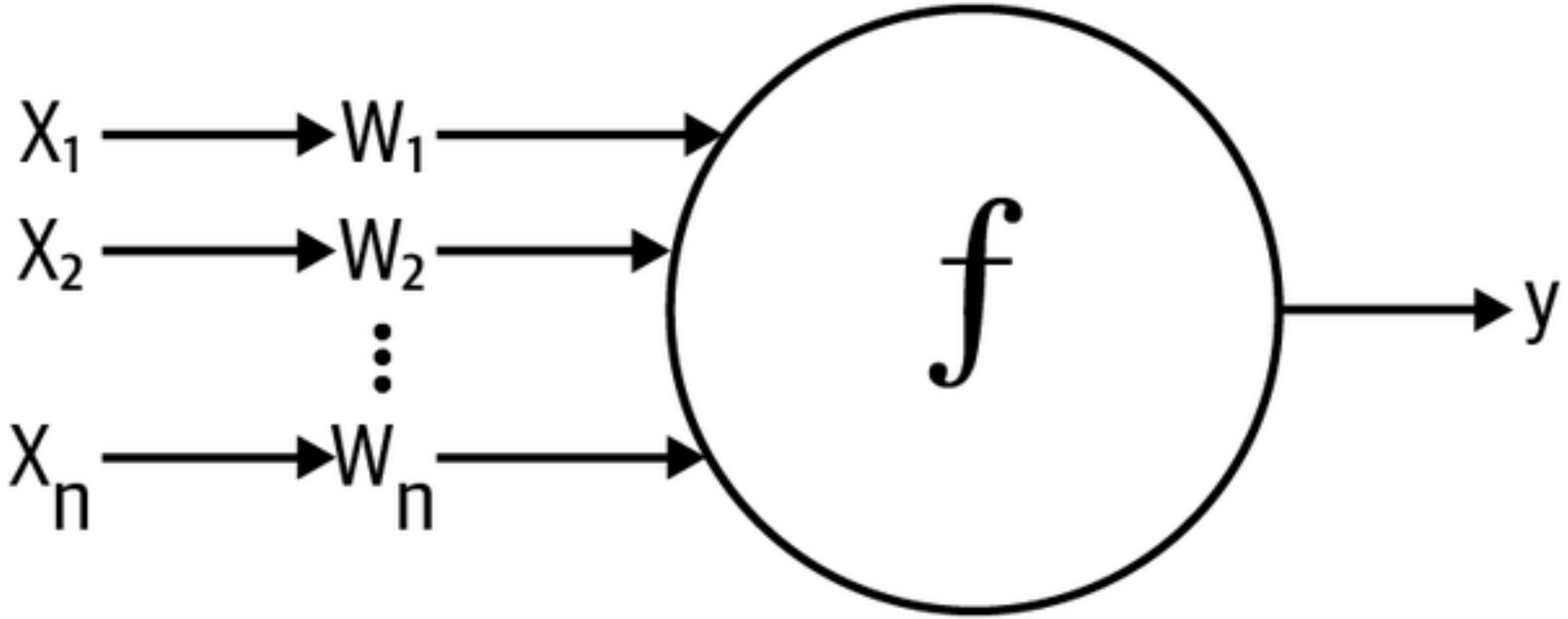
- Still believe that the names are terrible
  - Neural net, neuron, …



Terminology associated with neuron

- Translate the idea into more usable form
- Our artificial neuron takes in some number of inputs, $x_1, x_2, \dots, x_n$,each of which is multiplied by a specific weight, $w_1, w_2, \dots, w_n$
- These weighed inputs are summed to produce the *logit* of the neuron, $z = \sum_i w_i x_i + (bias)$
  - A bias is just a constant
- The logit is then passed through a function f to produce the output $y = f(z)$

Schematic for a neuron in a neural net



Formulation

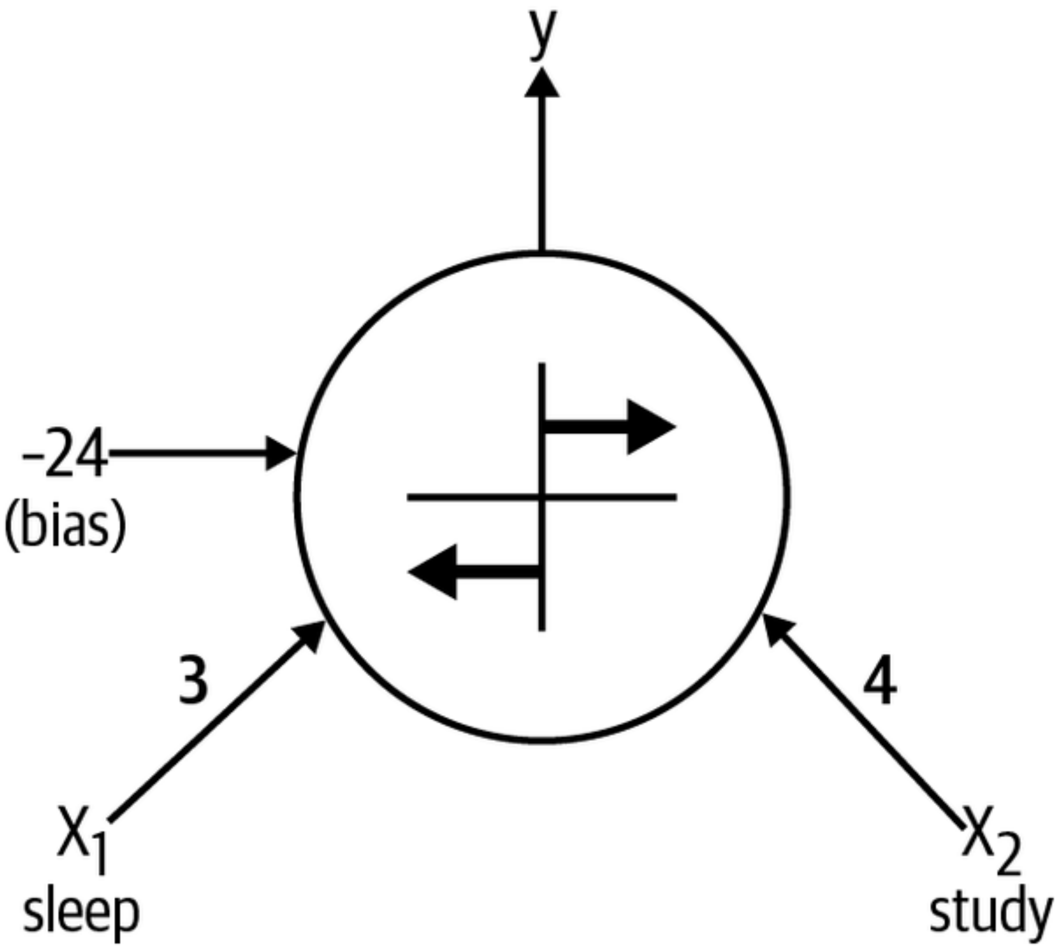
- Express its functionality in vector form
  - Input $\boldsymbol{x} = [x_1, x_2, \dots, x_n]$
  - Weights of the neuron $\mathbf{w} = [w_1, w_2, \dots, w_n]$
  - The output of the neuron $y = f(\boldsymbol{x} \cdot \boldsymbol{w} + b)$, where b is the bias term
- While this seems like a trivial reformulation, thinking about neurons as a series of vector manipulations will be crucial to how we implement them in software

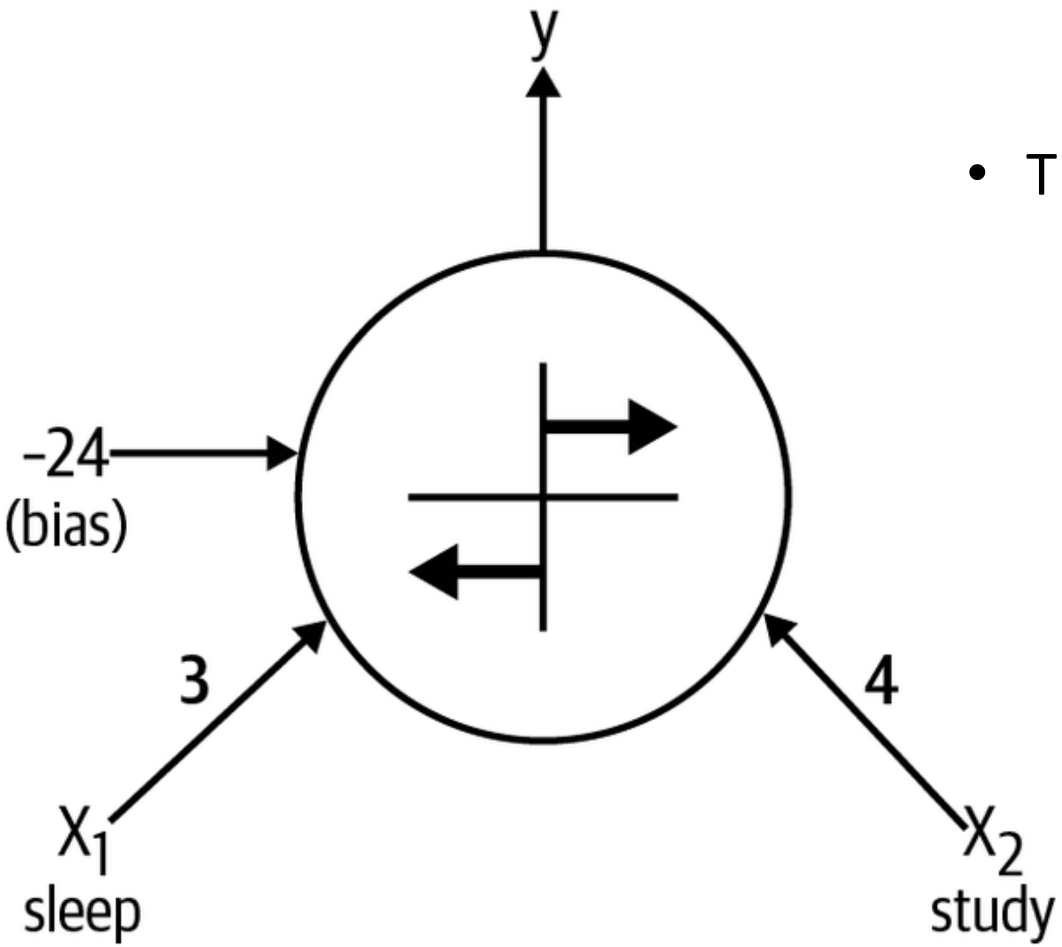
Expressing Linear Perceptrons as Neurons

- Talked about using machine learning models to capture the relationship

$$h(\mathbf{x}, \theta) = \begin{cases} -1 & \text{if } 3x_1 + 4x_2 - 24 < 0 \\ 1 & \text{if } 3x_1 + 4x_2 - 24 \geq 0 \end{cases}$$

- Here, we show that our model *h* is easily using a neuron


y
-24
(bias)
3
4
X_1
sleep
X_2
study



- The neuron has two inputs, a bias, and uses:

$$f(z) = \begin{cases} -1 & \text{if } z < 0 \\ 1 & \text{if } z \geq 0 \end{cases}$$

Discussion

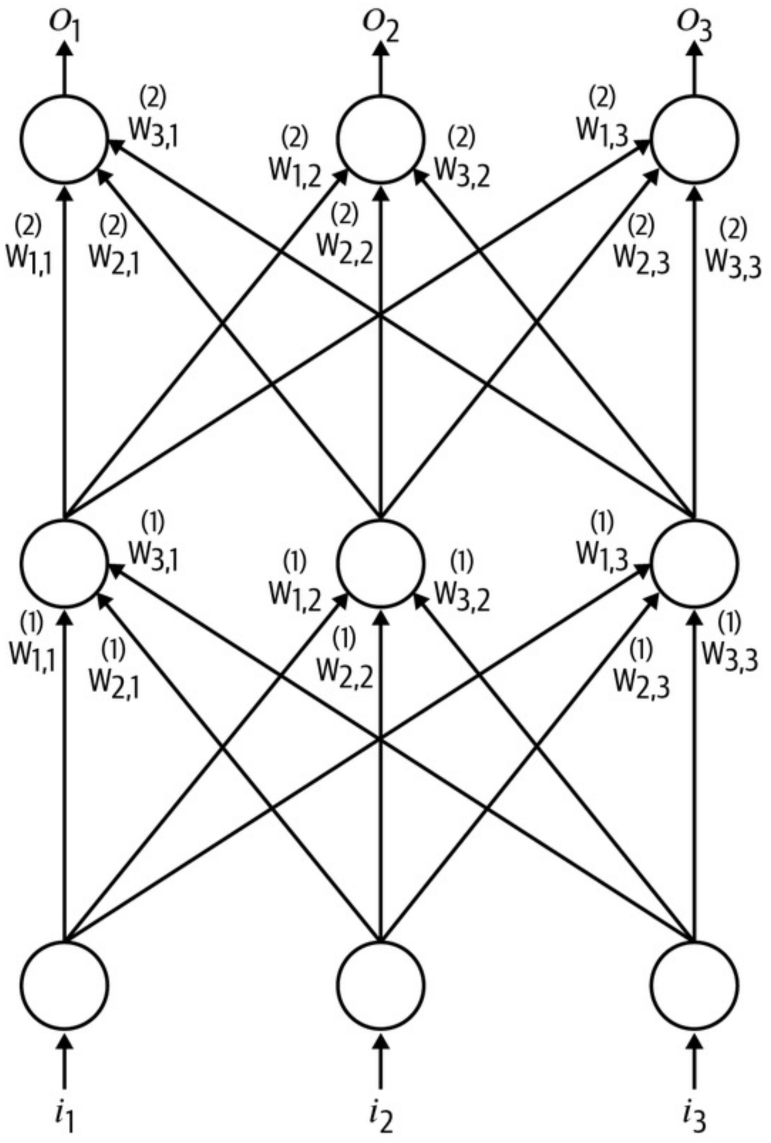
- It’s easy to show that our linear perceptron and the neuronal model are perfectly equivalent
- In general, it’s quite simple to show that singular neurons are strictly more expressive than linear perceptrons
  - Every linear perceptron can be expressed as a single neuron, but single neurons can also express models that cannot be expressed by any linear perceptron

Better together: network

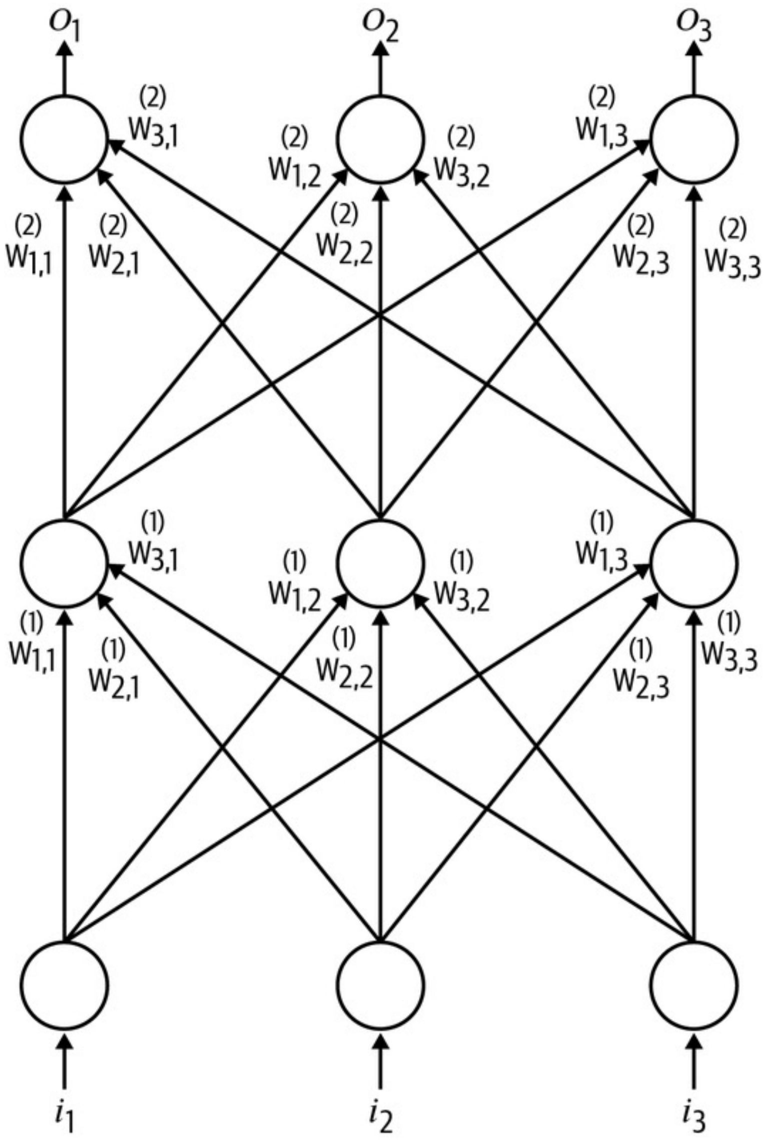
- Although single neurons are more powerful than linear perceptrons, they’re not enough
  - XOR problems
- Note that our brain is made of more than one neuron
- Consider the meaning of multiple neurons

Neurons are organized in layers

- Human cerebral cortex (the structure responsible for most of human intelligence) is made up of six layers
  - Information flows from one layer to another until sensory input is converted into conceptual understanding
  - I am just saying… please don't ask
- Borrowing from these concepts, we can construct an artificial neural network



- The bottom layer of the network pulls in the input data
- The top layers (output nodes) computes the final answer
- The middle layer(s) are called hidden layers
    - $w_{i,j}^{(k)}$ is the weight of the connection between i^{th} neuron in the k^{th} layer with the j^{th} neuron in the $k+1^{th}$ layer
- All the weights constitute our parameter vector θ



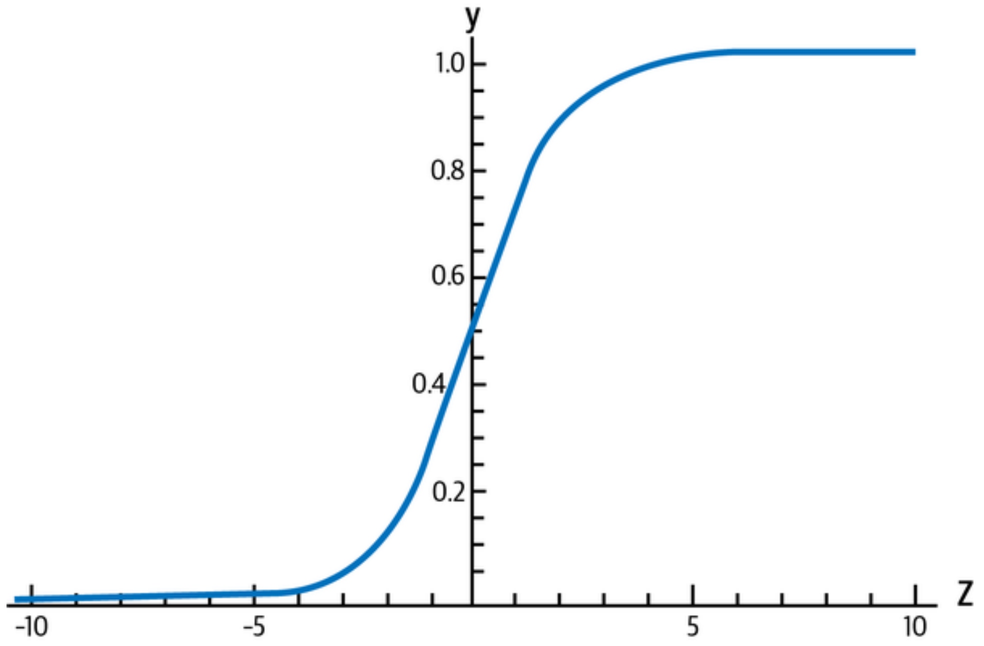
- These neural networks are called *feed-forward* networks
    - Most simple form
    - Connections traverse only from a **lower** layer to a **higher** layer
- No connections **between** neurons in the same layer

Note

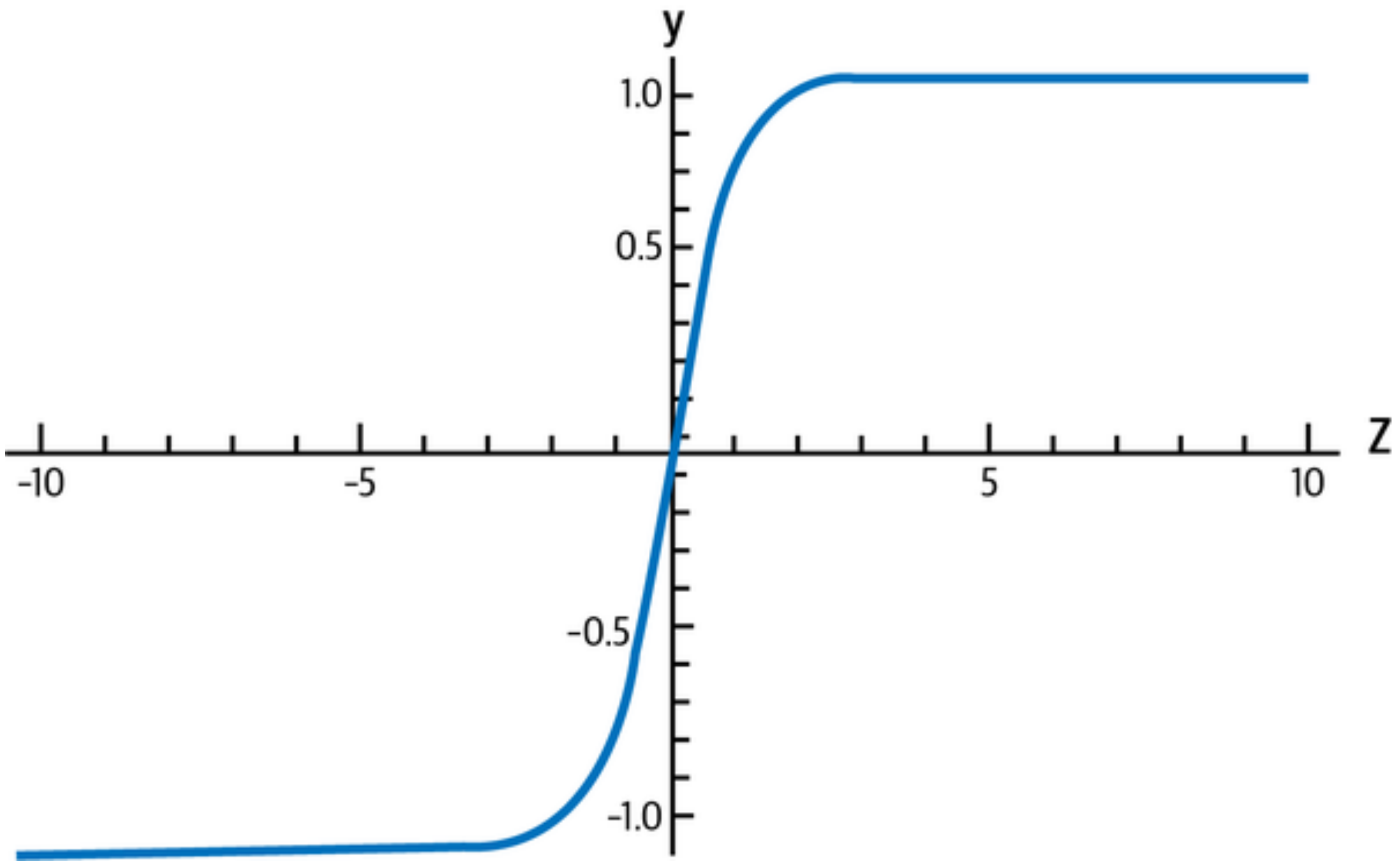
- Hidden layer is where most of the magic is happening when the neural net tries to solve problems
- Whereas we would previously have to spend a lot of time identifying useful features, the hidden layers automate this process for us.
- Oftentimes, taking a look at the activities of hidden layers can tell you a lot about the features the network has automatically learned to extract from the data
  - In fact, here we get the word embeddings

Sigmoid, Tanh, and ReLU Neurons

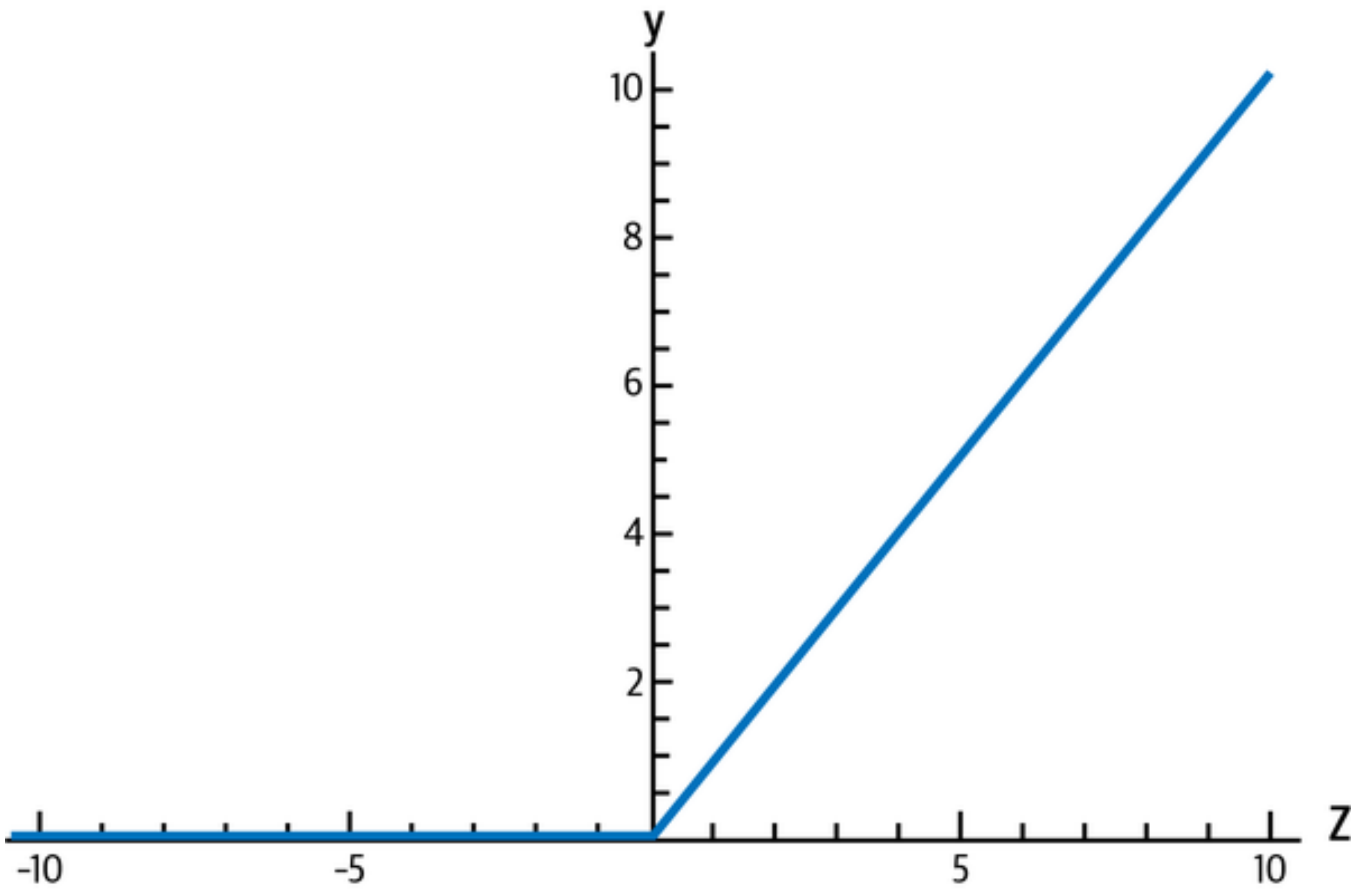
- Three major types of neurons are used in practice that introduce nonlinearities in their computations
- The sigmoid neuron uses the function $f(z) = \frac{1}{1+\exp(-z)}$



$$f(z) = \tanh(z)$$



Rectified Linear Unit: $f(z) = \max(0, z)$



Softmax Output Layers

- Oftentimes, we want our output vector to be a probability distribution over a set of mutually exclusive labels
  - For example, let's say we want to build a neural network to recognize handwritten digits from the MNIST dataset
  - Each label is mutually exclusive
- Using a probability distribution gives us a better idea of how confident we are in our predictions: $\sum_{i=0} p_i = 1$
- This is achieved by using a special output layer called a softmax layer
- Unlike in other kinds of layers, the output of a neuron in a softmax layer depends on the outputs of all the other neurons in its layer

Softmax Output Layers

- This is because we require the sum of all the outputs to be equal to 1
- Letting z_i be the logit of the i^{th} softmax neuron, we can achieve this normalization by setting its output to:
  - $y_i = \frac{\exp(z_i)}{\sum_j \exp(z_j)}$
- A strong prediction would have a single entry in the vector close to 1, while the remaining entries would be close to 0
- A weak prediction would have multiple possible labels that are more or less equally likely
- Most popular in language models

Keras

- Keras [example]