Lecture 2

Goals

- what is a neural net?
- intuition on structure
- weights, biases + activations
- neural nets as universal approximator of continuous functions

Image Recognition

Trivial for humans ⇒ challenging for machines

- can identify different "forms" of the # 3

Computer sees value of every single pixel

- Takes in giant matrix ⇒ tries to recognize what # it is

Neural net can help us do these types of recognition

Why Layers?

* Just talking about plain neural network*
  - multi-layer perceptron

Main ingredients

neurons: things that hold a number
- takes in each computer pixel as value the neuron holds

Input layer

Pixel value

Hidden Layers

} Probability image belongs to a certain digit

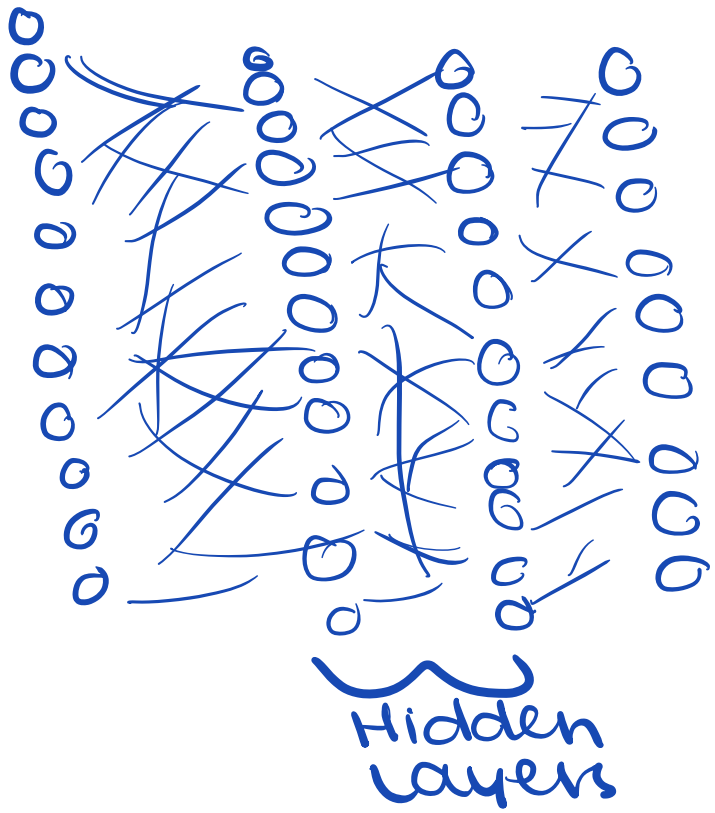
a neuron = Prob the image belongs to 9

Take maximum Probability as our prediction

# of neurons = # of pixels

Layers used to hold info we care about

No reverse neuron connection, all directed



can have any # of neurons + hidden layers

Dissecting Image Recognition

9 → O + l } looks like a 9 because we see components

8 → O + O

4 → \ + — + |

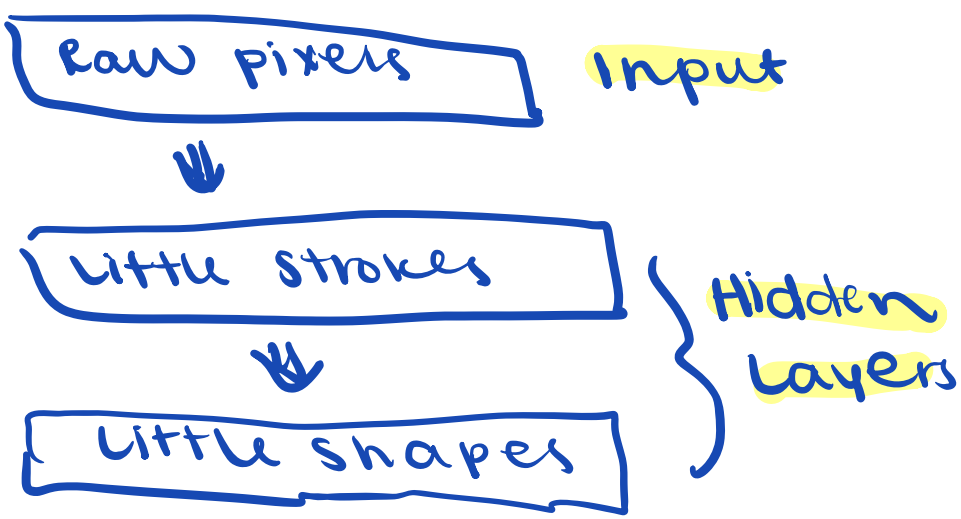
Our brain recognizes the digit by looking for essential features in their relative location

Hidden layers may pick up small patterns in digits to make predictions

We can ever further dissect the digit components

0 → (+ (+ — +) +)

this drawing sucks but you get the idea



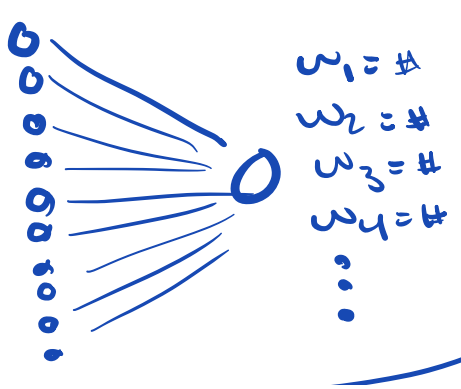
Layers → complex tasks can be decomposed to different levels of granularity

Interpretability is still an open question in AI

- we don't know how to interpret neural nets or if its using smaller pieces at every single layer

Connections in neural nets

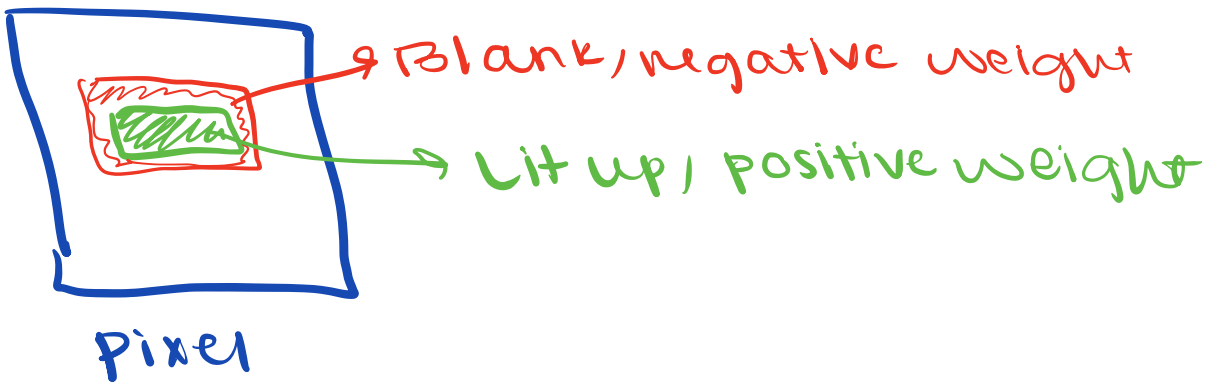
passes on info via weights



want neuron to light up when the pattern exists

Also need to think about region outside path

want it to be dark = use negative weights



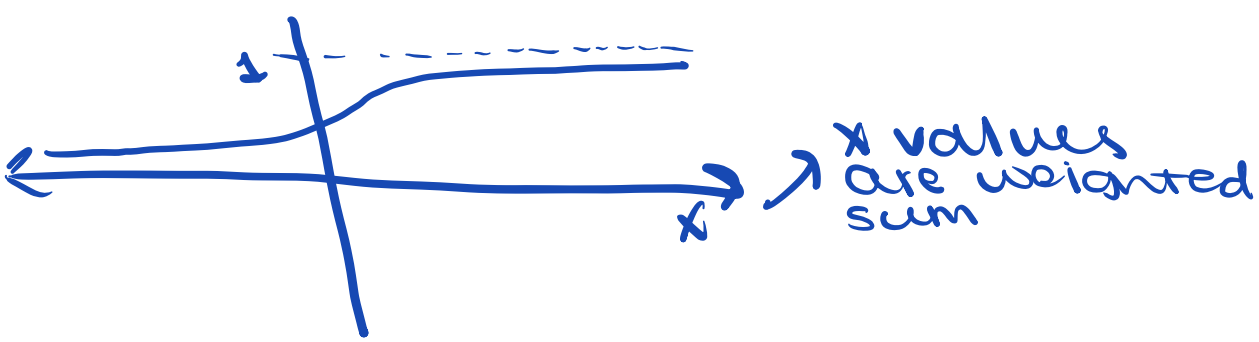
Activation Function

Activation should be in range 0 to 1

Convert weighted combination to [0,1] w/ activation function

- squish #'s into range

$$\sigma(x) = \frac{1}{1+e^{-x}}$$] Sigmoid Function



Sigmoid lets us capture nonlinear structures

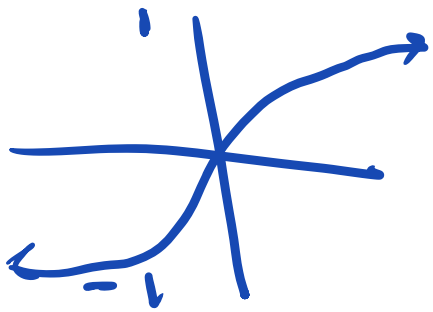
Sigmoid is just one activation function
↳ tells us how positive or negative

Tanh (Hyperbolic Tangents)

maps -1 to 1

very small → -1

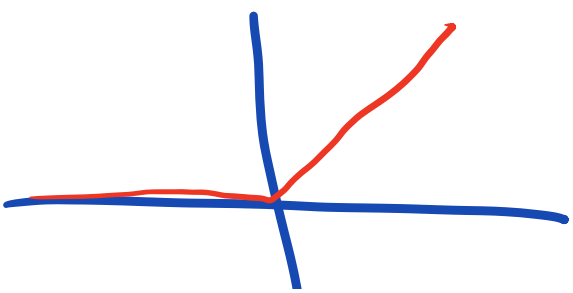
very large → 1



Rectified Linear unit

$X \to \max(0, x)$

- 0 out negative component
- preserves magnitude



1

can activation functions be linear?

↳ No, makes neural net only one layer

↳ doesn't add to complexity or fluxibility

Bias in Neural Nets

Bias for inactivity?

$\sigma(w_1a_1 + w_2a_2 \dots w_na_n - 10)$

use bias so it doesn't activate when sum > 0

Bias, only activates when weighted sum is > 10

every neuron has a bias

Learning in Neural Nets

Learning: finding right weights + biases for every neuron + layer

Compact Notation

$$a_0^{(1)} = \sigma(w_{0,0}a_0^{(0)} + w_{0,1}a_1^{(0)} + w_{0,2}a_2^{(0)} + \dots + w_{0,n}a_n^{(0)} + b_0)$$

σ: sigmoid

b_0: bias

↳ looking at 1 neuron

a_0 = 0^{th} layer

↳ weighted input + bias

↳ into activation function

we have one equation for each node

$$\begin{bmatrix} w_{0,0} & w_{0,1} & \dots & w_{0,n} \\ w_{1,0} & w_{1,1} & \dots & w_{1,n} \\ \vdots & \vdots & \ddots & \vdots \\ w_{k,0} & w_{k,1} & \dots & w_{k,n} \end{bmatrix} \begin{bmatrix} a_0^{(0)} \\ a_1^{(0)} \\ \vdots \\ a_n^{(0)} \end{bmatrix}$$

weights

one input vector

can rewrite equation using matrix

product of row of weights + column of inputs

$$a_1^{(1)} = w_{10}a_0^{(0)} + w_{11}a_1^{(0)} + \dots w_{1n}a_n^{(0)}$$

different weights, same set of inputs

$$\sigma\left(\begin{bmatrix} w_{00} & w_{01} & \dots & w_{0n} \\ w_{10} & w_{11} & \dots & w_{1n} \\ \vdots & \vdots & \ddots & \vdots \\ w_{k0} & w_{k1} & \dots & w_{kn} \end{bmatrix}\begin{bmatrix} a_0^{(0)} \\ a_1^{(0)} \\ \vdots \\ a_k^{(0)} \end{bmatrix} + \begin{bmatrix} b_0 \\ b_1 \\ \vdots \\ b_k \end{bmatrix}\right)$$

$k \times n$ $\quad n \times 1$ $\quad k \times 1$

$$a' = \sigma(Wa^{(0)} + b)$$

This notation also makes coding easier!

- languages optimized for vectorized computation

Neurons As Functions

neuron doesn't hold "one number"
↳ different inputs will produce different outputs

Neural Network ≈ big complicated function

Training Neural Nets

Start w/ initial net w/ random values for weights
↳ typically get incorrect predictions
↳ calculate how far prediction is from true value

↳ send signal back to update weights

Neural Nets As Universal Approximator

Can approx. any continuous function w/ a linear combo of translated/scaled ReLU functions

- single hidden layer MLP
- But may require infinite neurons in the layer

$$f(x) = \sigma(w_1 x_1 + b_1) + \sigma(w_2 x_2 + b_2) + \ldots$$

- Apply weights to every input + biases
- Apply sigmoid function to each output

$$\sigma(x) = \max(0, x)$$

- use small segments to approx continuous function

Approx any Boolean, classification or regression function to arbitrary precision

we need to specify how wide & big we want hidden layers to be

- Tune # of layers & neurons as iterative process

neurons can ↑ from layer to layer

* Neural net can represent any function if it has sufficient capacity
  - ↳ Not all architecture can represent any function