Artificial neural networks: The fundamentals

Julie Midroni

Presenting the Problem

- Getting a function to spit out the right vector if we feed it the right numerical input
- Seems straightforward, but...

Presenting the Problem

- I may not know the function, only certain points

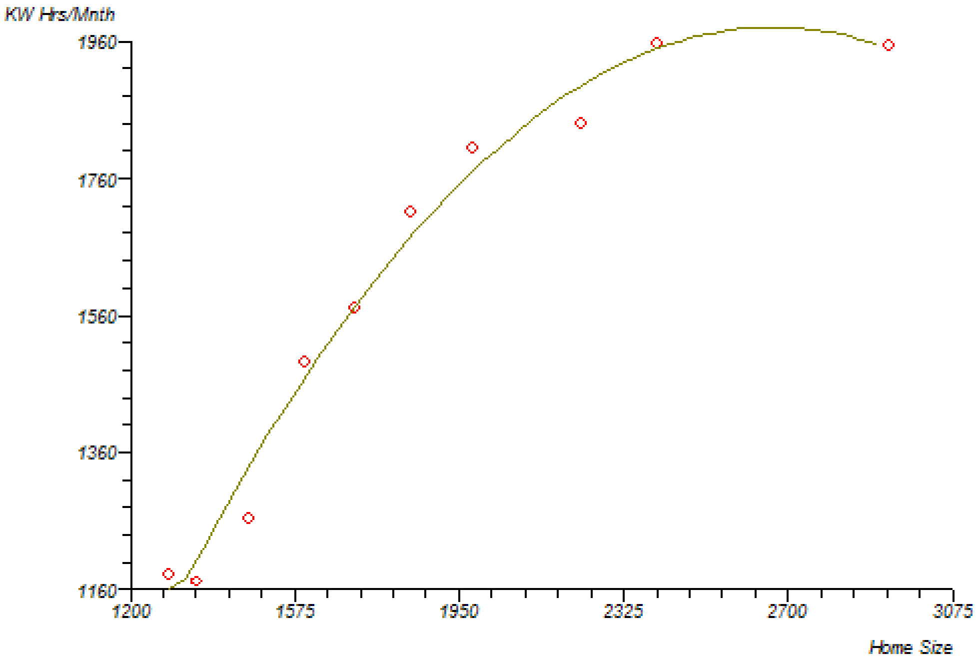
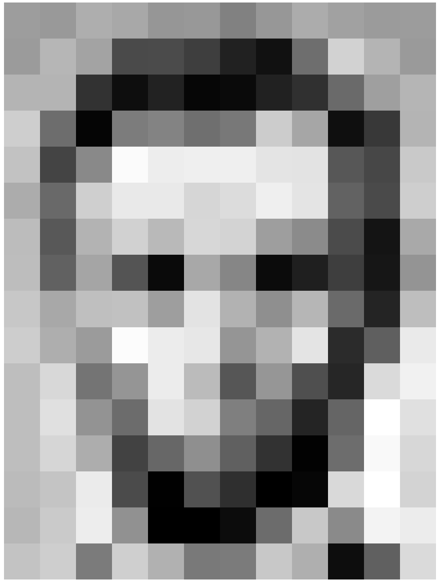
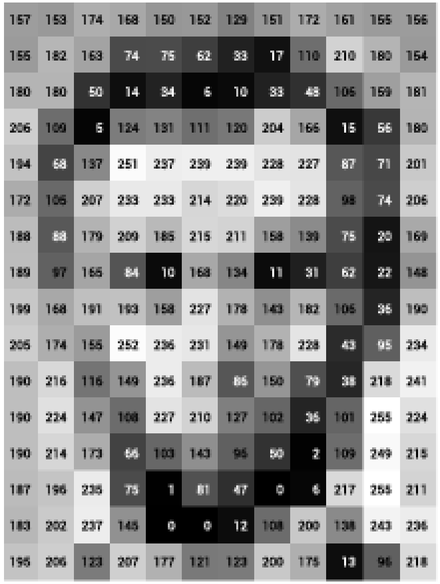


Image credit: [1]

Presenting the Problem

- An image is an array of numbers (classification problem!)





157	153	174	168	150	152	129	151	172	161	155	156
155	182	163	74	75	62	33	17	110	210	180	154
180	180	50	14	34	6	10	33	48	106	159	181
206	109	5	124	131	111	120	204	166	15	56	180
194	68	137	251	237	239	239	228	227	87	71	201
172	105	207	233	233	214	220	239	228	98	74	206
188	88	179	209	185	215	211	158	139	75	20	169
189	97	165	84	10	168	134	11	31	62	22	148
199	168	191	193	158	227	178	143	182	106	36	190
205	174	155	252	236	231	149	178	228	43	95	234
190	216	116	149	236	187	86	150	79	38	218	241
190	224	147	108	227	210	127	102	36	101	255	224
190	214	173	66	103	143	95	50	2	109	249	215
187	196	235	75	1	81	47	0	6	217	255	211
183	202	237	145	0	0	12	108	200	138	243	236
195	206	123	207	177	121	123	200	175	13	96	218

Image credit: [2]

Presenting the Problem

- Even words can be converted to numbers
  - This is how you make chat bots!

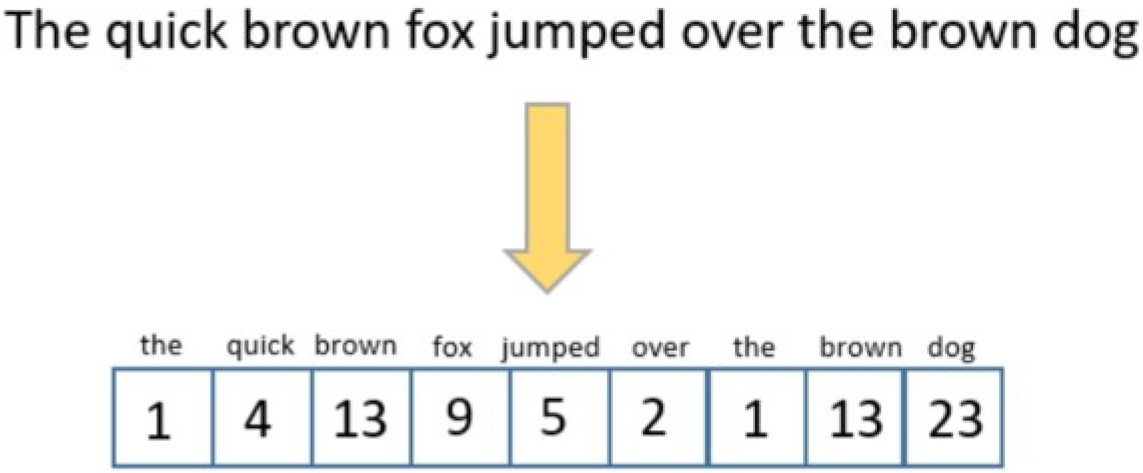
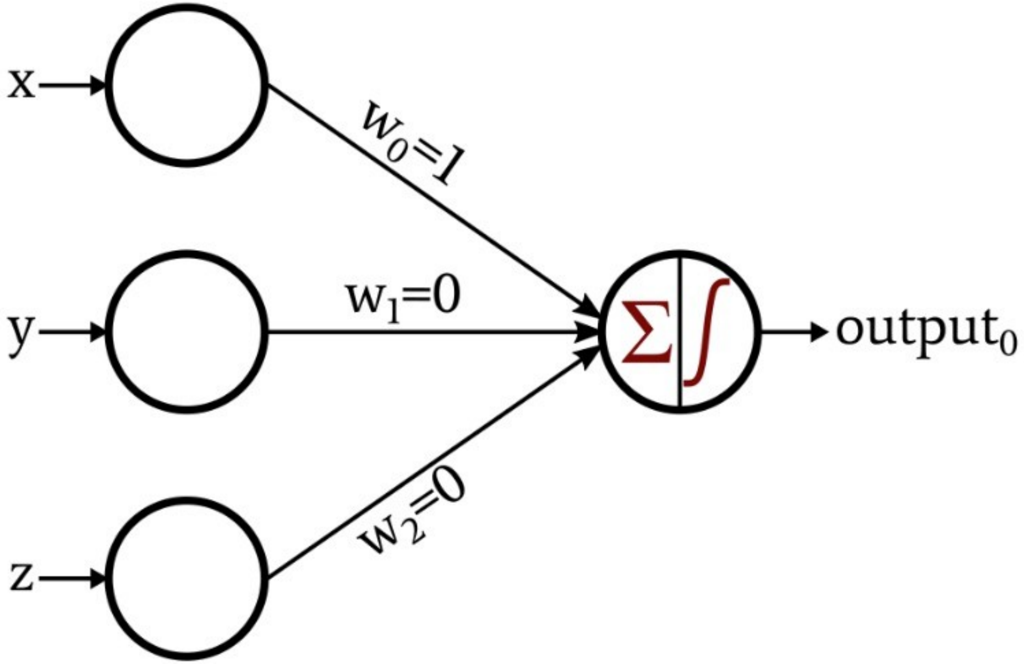


Image credit: [3]

[3]

Artificial Neurons

- Building blocks of neural networks
  - Inputs
  - Weights
  - Summation
  - Activation function
    - Ex: threshold, sigmoidal

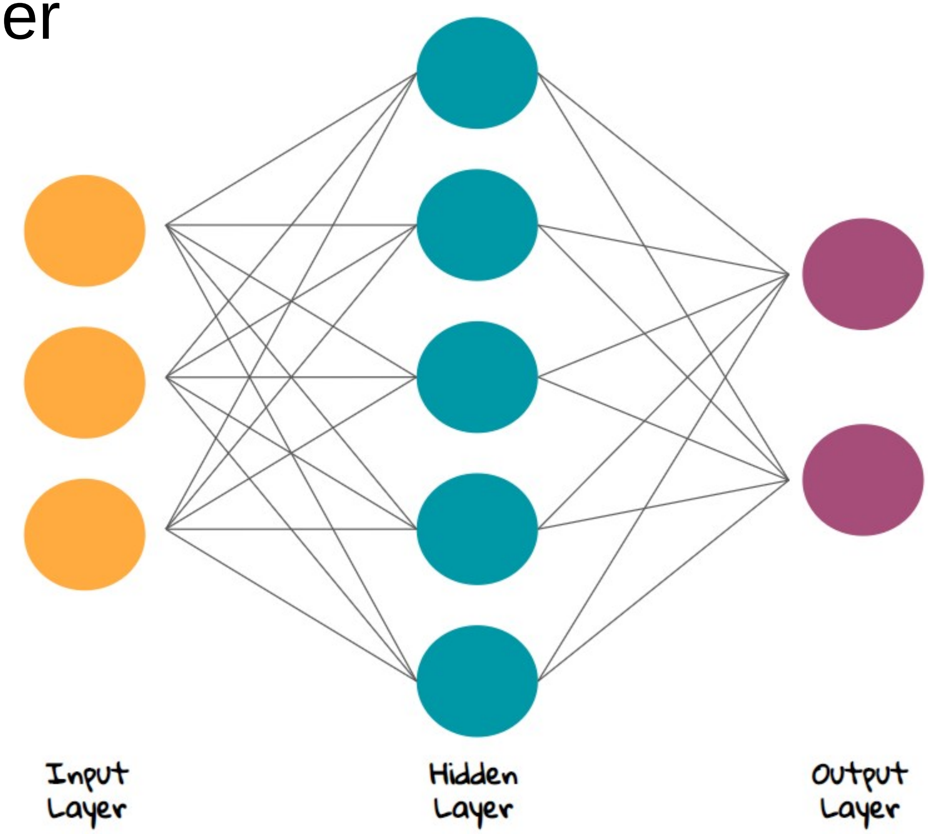


[4]

Image credit: [4]

Artificial Neural Networks

- Chain multiple neurons together
  - Series of linear and nonlinear transformations allows for input to be ‘converted’ to correct output



[5]

Image credit: [5]

Forward Propagation Walkthrough

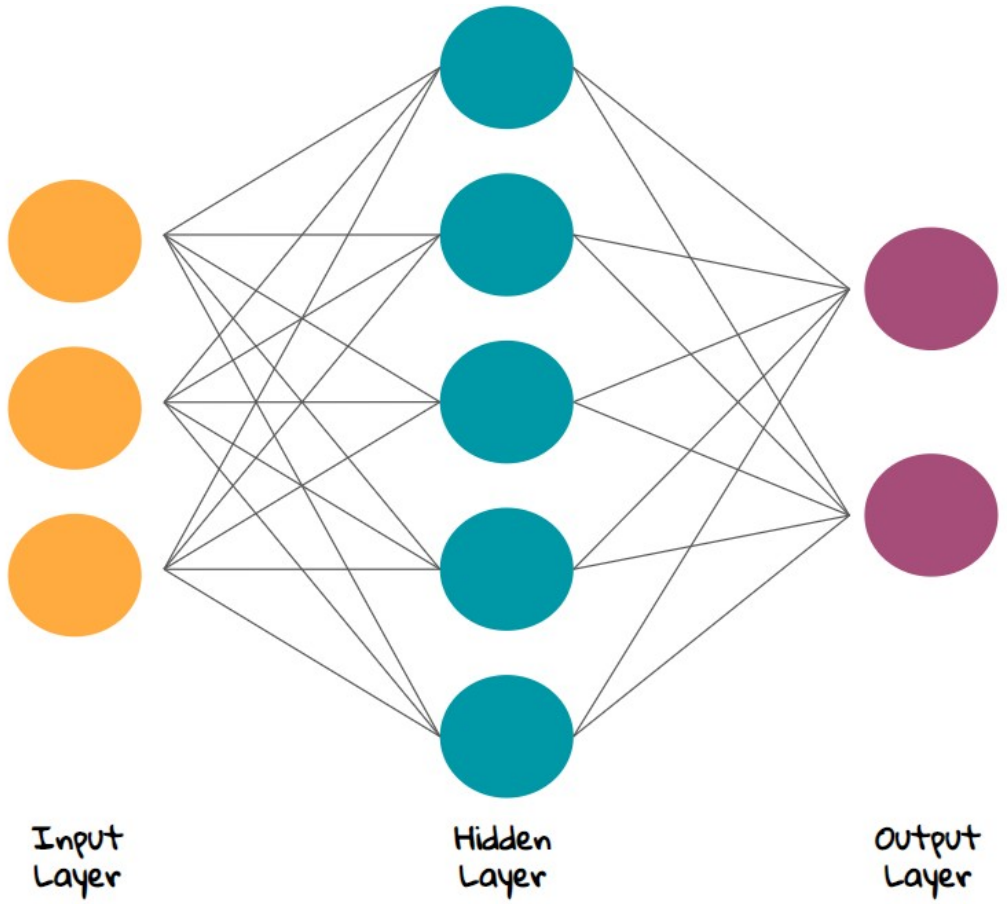


Image credit: [5]

DEF: *Two sets of points A and B $\subseteq R^n$ are linearly separable if there exist n + 1 points w_0 , ..., w_n such that for any point a in A,* $\sum_1^n w_i a_i < w_0$ *, and for any point b in B,* $\sum_1^n w_i b_i > w_0$

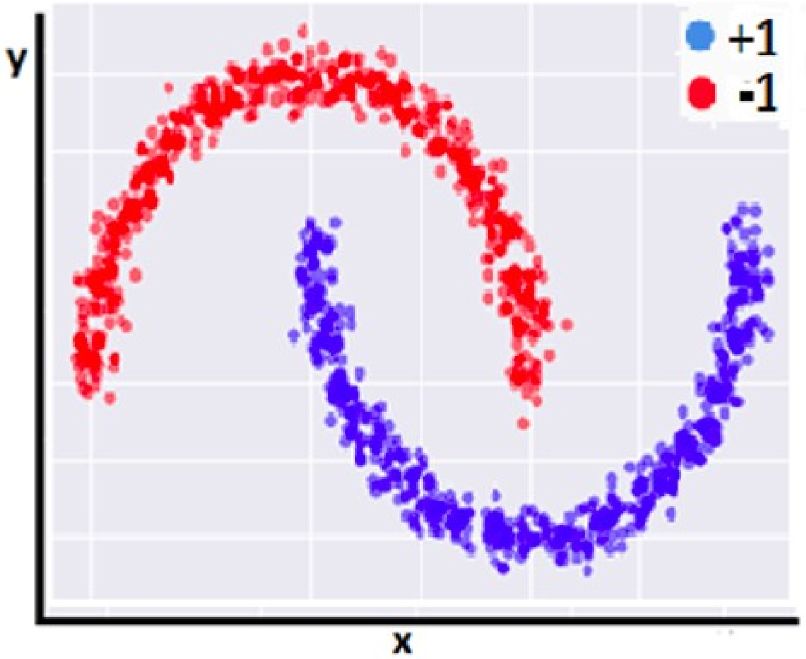
- Effectively, two sets are linearly separable if there exists some hyperplane that separates the two sets

[6]

Why this is necessary: Linear separability

- What an individual node without an activation function does is place an input above/below a hyperplane

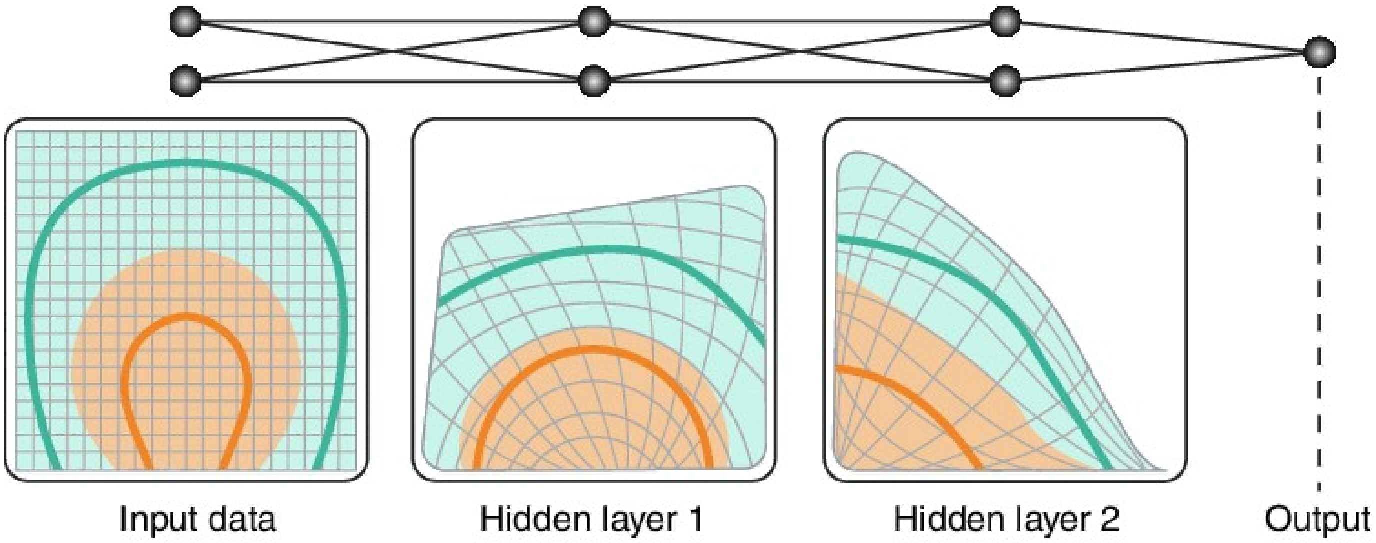
- What if data is not linearly separable?
  - Drawing a line will not help me separate these two groups
  - So the network won’t be able to tell the difference in its output!



[6]

Image credit: [7]

- Multiple nodes, layers, and nonlinear activation transform our data to something linearly separable



[8]; Image credit: [8]

Part 1 Summary: Artificial Neural Networks Work!

- Given an input, so long as the weights are correct, an ANN will return an output with a numerical value that either:
  - Represents the ‘class’ of the input
  - Is the image of the input, as transformed by some unknown (approximated) function

REGARDLESS OF WHETHER OR NOT THE INPUT CAN BE LINEARLLY SEPARATED INTO OUTPUT CATEGORIES

[6]

The next problem: Weights?

- How do we determine weights?
  - Training!

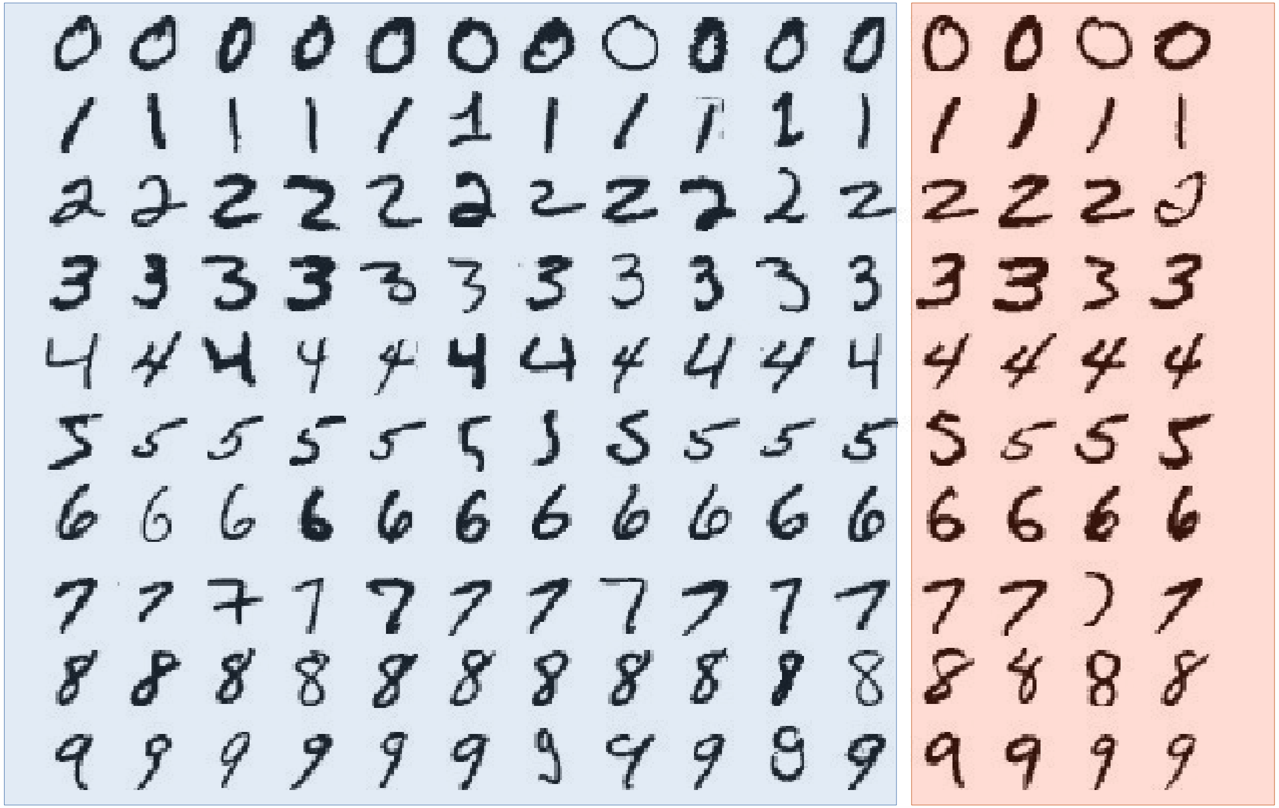


Image credit: [9]

Backpropagation

- Backpropagation is how we train ANNs
  - Weight-update algorithm

- Basic concept
  1) Forward propagate training data
  2) Calculate error
  3) Update weights to minimize error
  4) Repeat

[10]

Backpropagation

- STEP 1: Forward propagate the data
  - This is just a normal forward pass
  - We get a result, which may or may not be ‘correct’

[10]

Backpropagation

- STEP 2: Calculate error

  DEF: for some piece of training data x, let y_t be the expected (correct) output, and let y_e be the actual output of the network

- Define some differentiable error function $E(y_t , y_e)$ that represents the error between the expected and actual outputs
  - We call the error of the output the “loss”
  - We want to **MINIMIZE LOSS**

[10]

Backpropagation

- STEP 3: Update weights to minimize loss
  - Use the backpropagation algorithm, which employs the chain rule

  - Recall the gradient of a function is a vector of the partial derivatives of the function wrt each component
    - If the gradient is positive in a direction, the function is increasing in that direction

[10]

Backpropagation

- Remember, we want to minimize E
  - Get the gradient of E to zero

- E is a function of the output, and thus a function of the input and the weights $\{w_1, \ldots, w_k\}$
  - If $\frac{\partial E}{\partial w_i}$ is positive, E is increasing as w_i does, so we want to decrease w_i

[10]

Backpropagation

- Weight update rule: $w_i = w_i - (l \frac{\partial E}{\partial w_i})$

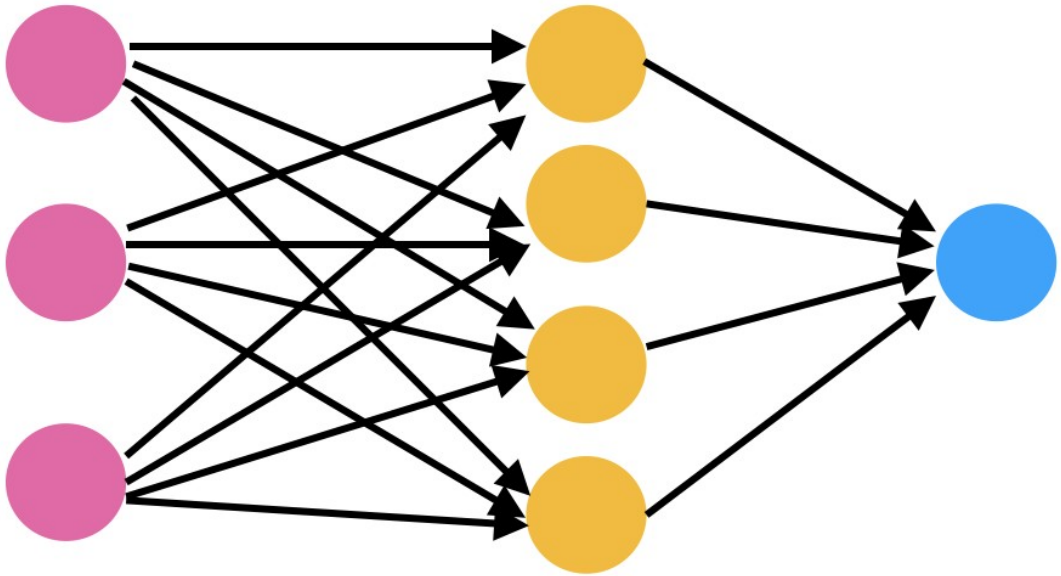
  l is the learning rate of the network

[10]

Backpropagation

- How do we find the partial derivative wrt w_i? Chain rule!

- E is a function of y_e
- y_e is a function of the layer's input, z
- z is a function of the previous layer's output and the weights



[10]

Image credit: [11]

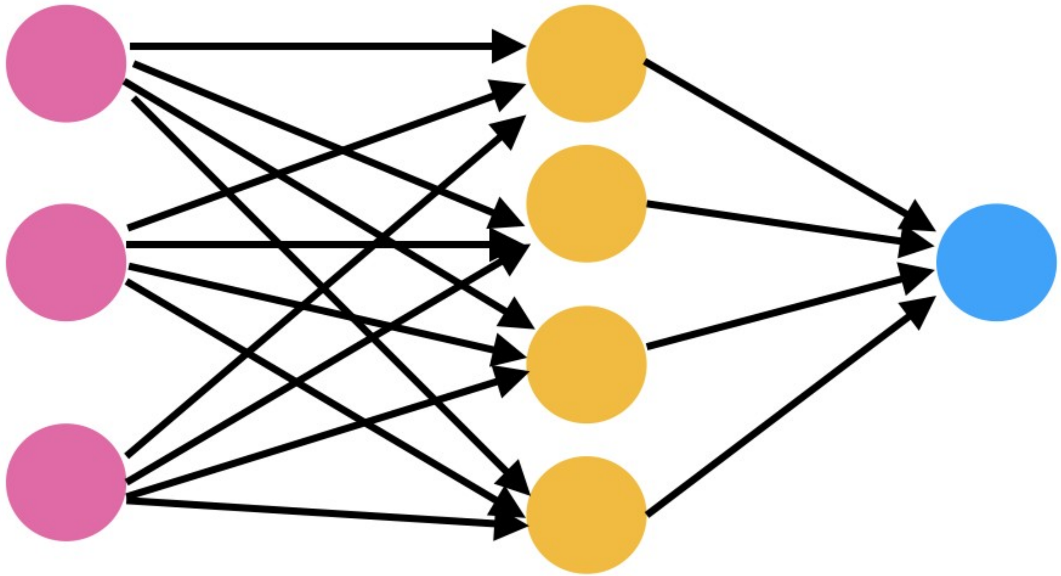
Backpropagation

- How do we find the partial derivative wrt w_i? Chain rule!

- E is a function of $y_{e,i}$ $\frac{\partial E}{\partial y_{e,i}}$

- y_e is a function of the layer's input, z $\frac{\partial y_{e,i}}{\partial z}$

- z is a function of the previous layer's output and the weights $\frac{\partial z}{\partial w_i}$



[10]

Image credit: [11]

Backpropagation

- How do we find the partial derivative wrt w_i? Chain rule!

$$\frac{\partial E}{\partial w_i} = \frac{\partial E}{\partial y_{e,i}} \frac{\partial y_{e,i}}{\partial z} \frac{\partial z}{\partial w_i}$$

[10]

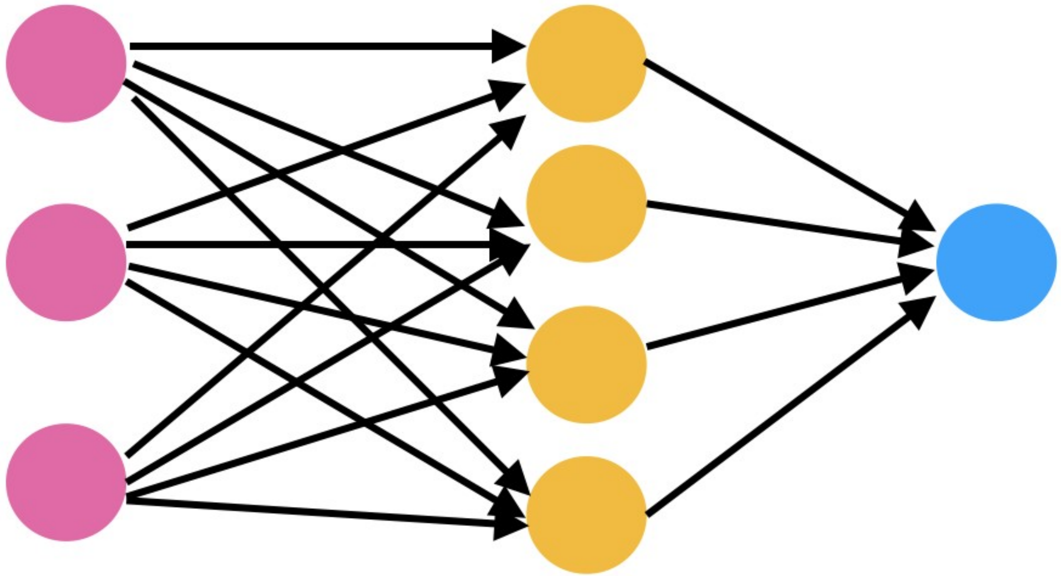


Image credit: [11]

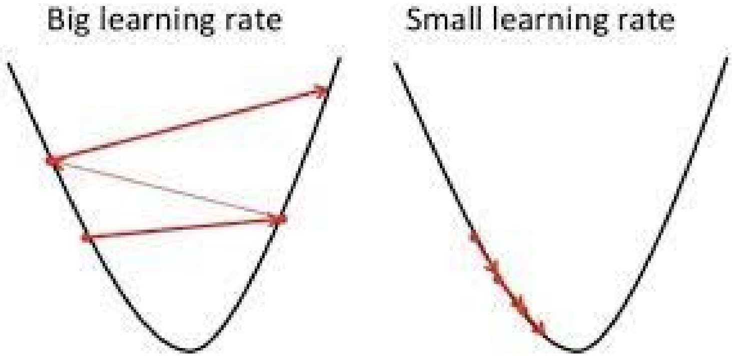
Backpropagation

- In summary:
  1) Forward pass
  2) Calculate loss
  3) Backpropagate and weight update
  4) Rinse and repeat until loss has been minimized

[10]

Backpropagation

- So that’s how you train a neural net!
  - There are some other tweaks you can make
  - There are some issues
    - Local minima
    - Fine-tuning learning rate



[10]

Image credit: [12]

Intuitively, what’s happening?

- By adjusting weights, ANNs pick out which inputs are important in certain circumstances
  - The weights are chosen to notice patterns in inputs, and produce different outputs with the same weights, depending on the presence/absence of certain patterns (features)
  - Far more powerful than a simple regression
  - Robust to small changes in the input, so long as general patterns are present (ie: different species of cats are still cats)

[10]

Time for a quick demo!

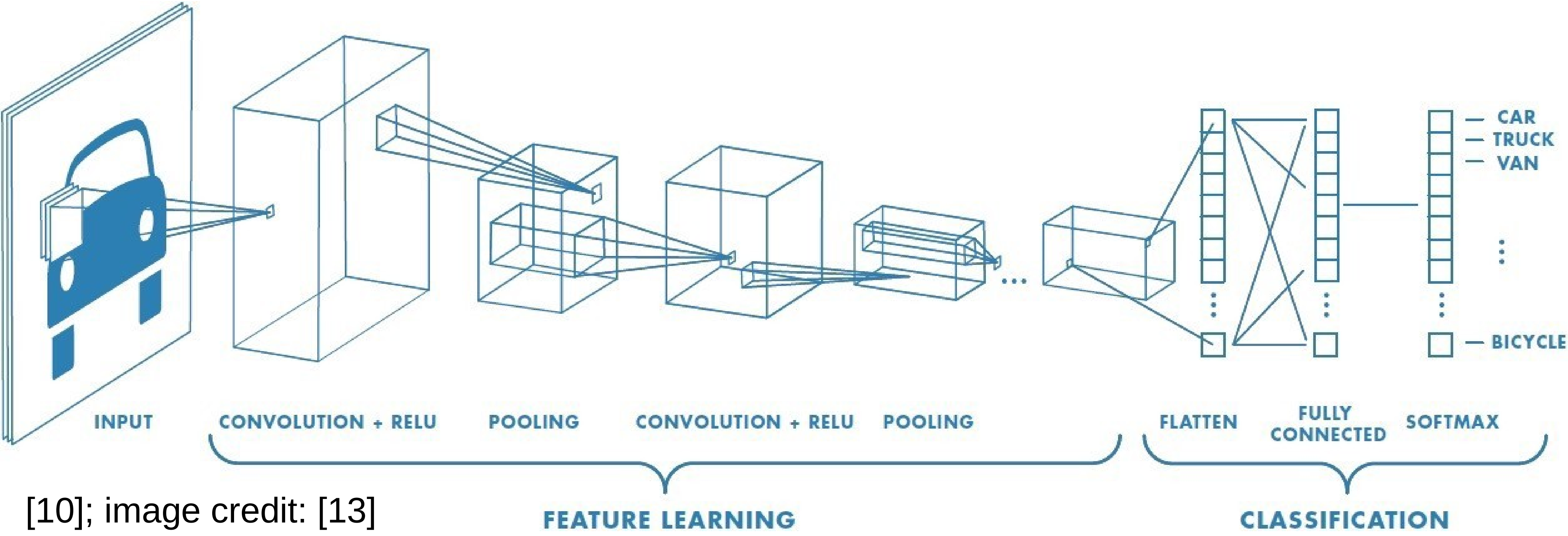
Advanced ANNs

- Recurrent ANNs allow for time-dependent data
  - Output of previous timestep becomes an input
  - Good for chat bots

[10]

Advanced ANNs

- Convolutional neural networks are good for image classification



[10]; image credit: [13]

References

[1] *Polynomial Regression.* StatsDirect Limited, 2000. https://www.statsdirect.com/help/regression_and_correlation/polynomial.htm

[2] *Tutorial 1: Image filtering*. Yeung, S, n.d. https://ai.stanford.edu/~syyeung/cvweb/tutorial1.html

[3] *Text encoding: A* review. Silipo, R. and Melcher, K., 2019. https://www.kdnuggets.com/2019/11/text-encoding-review.html

[4] *How to train a basic perceptron neural network*. Keim, R., 2019. https://www.allaboutcircuits.com/technical-articles/how-to-train-a-basic-perceptron-neural-network/

[5] *Basics for beginners – Neural networks – Part 2*. Murthy, P. K., 2021. https://dockship.io/articles/608580a840249767deda2419/basics-for-beginners---neural-networks---part-2

[6] *10.1 – When data is linearly Separable.* The Pennsylvania State University, 2021. https://online.stat.psu.edu/stat508/

[7] *Using a linear model to deal with nonlinear dataset.* Kaushik, S., 2019. https://medium.com/@sachinkun21/using-a-linear-model-to-deal-with-nonlinear-dataset-c6ed0f7f3f51

[8] Esteva, A., Robicquet, A., Ramsundar, B., and Kuleshov, V., A guide to deep learning in healthcare. Nature Medicine 25(1) (2019). dx.doi.org/10.1038/s41591-018-0316-z

[9] Cortes, C. And Y. LeCun., MNIST handwritten digit database. (2010). http://yann.lecun.com/exdb/mnist/

[10] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning* (MIT Press, 2016). https://www.deeplearningbook.org/

References

[11] *Understanding the structure of neural networks. Logan, S.*, 2017. https://becominghuman.ai/understanding-the-structure-of-neural-networks-1fa5bd17fef0

[12] *Learning rate in machine learning*. Educative, n.d. https://www.educative.io/edpresso/learning-rate-in-machine-learning

[3] *A comprehensive guide to convolutional neural networks – the ELI5 way*. Saha, S., 2018. https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53