CSC 236 HW 2

Check in period: 6/8 - 6/13

About

Please see the guide to hw, and guide to check ins for information and tips for the HW and the check ins!

These homeworks are on lectures and tutorials 1-4 inclusive.

1 Song Order

In this problem, you've been hired as a consultant for guitar player extraordinaire - Andy McKee. You've been asked to help him figure out the 'best' song order for a live gig. Here is some more context:

A guitar has six strings, each of which can be tuned to one of 88 musical pitches[1]. The strings are usually tuned to specific pitches known as 'standard tuning.' However, Andy uses many non-standard tunings and, for this problem, assume that every song he plays is in a different tuning. See here for an example of Andy tuning his guitar.

Here is an example. In standard tuning, strings 1 through 6 are tuned to the pitches 44, 39, 35, 30, 25, and 20, respectively. Strings 1 through 6 are tuned to 42, 37, 35, 30, 25, and 18 in one altered tuning. To get from standard tuning to this altered tuning, you'd tune the first, second and sixth strings down by 2 pitches each.

Let $\mathrm{Tunings}$ be the set of possible guitar tunings. Let $\mathrm{Songs} \subseteq \mathrm{Tunings}$ be the subset of $\mathrm{Tunings}$ that Andy would like to play for the show. Let S be standard tuning and assume $S \in \mathrm{Songs}$. Andy would like you to find the ordering of songs in Songs such that

- The guitar starts and ends in standard tuning S.
- The total amount of 'shifting' required is minimized.

Question 1a. Model this problem as one of the graph problems we studied in class. In particular, mathematically formalize the definitions of tunings, and fully specify the vertex set, the edge set, and edge weights (if any). Then, explain how a solution to the graph problem corresponds to the optimal song ordering described above.

Hint: define a notion of the cost of moving from tuning x to tuning y.

Let $\mathrm{Tunings} = [88]^6$ I.e. identify each tuning with a 6-tuple $(x_1, ..., x_6)$ where x_i denotes the pitch of each strings in a particular tuning.

Let $V = \mathrm{Songs} \subseteq \mathrm{Tunings}$, and E be the set of all possible edges between vertices in V. Let $G = (V, E)$.

We'll now define a notion of cost for moving between tuning x and tuning y. We'll encode this as our weight function $w : V \to \mathbb{R}$.

$$w((x_1, ..., x_6), (y_1, ..., y_6)) = \sum_{i=1}^{6} |x_i - y_i|.$$

[1]In reality, the set of possible pitches for each string is way smaller, but this assumption is OK for this problem since we restrict our attention to only the tunings that Andy uses

Note that $|x_i - y_i|$ is the number of pitches by which we need to tune string i, so the sum over all strings is the total amount we need to tune to move between tuning $(x_1, ..., x_6)$ and tuning $(y_1, ..., y_6)$. Note the absolute value - this is important since tuning a string up and tuning a string down both contribute to the cost. Without the absolute value, one direction would have negative cost!

Then we claim that finding the ordering of songs in Songs that starts and ends in standard tuning S and minimizes the total amount of shifting is given by the solution to the Traveling Salesman Problem on G, starting at vertex S, with weight function w.

Let $C = (S = s_1, ..., s_n = S)$ be the solution to the TSP on G. Since C is a cycle, we start and end in standard tuning, S. Furthermore, since C is Hamiltonian, every tuning is found in the cycle (every song is played). Finally, since C is the Hamiltonian that minimizes the total weight, and we encode the cost of moving from one tuning to the next in the weight function, C corresponds to the sequence of tunings that minimizes the total amount of shifting.

Note: You can define the edge weights however you want, as long as you justify it!

2 Trees

Question 2a. Let G be an undirected graph. Show that if G is maximally acyclic, then G is a tree.

Let $G = (V, E)$ be a maximally acyclic graph. We want to show that G is a tree. Since G is acyclic by assumption, we just need to show that G is connected.

Let $u, v \in V$ be any two vertices, we'll show that there is a path from u to v. If $\{u, v\}$ is an edge in G, then clearly, there is a path from u to v (namely the one consisting of just that edge).

Otherwise, $\{u, v\}$ is not an edge. Since G is maximally acyclic, adding the edge $\{u, v\}$ to G creates a cycle. Since G was acyclic to begin with, the cycle contains the edge $\{u, v\}$. Let $C = (u = u_1, v = u_2, ..., u_k = u)$. Then $P = (u = u_k, u_{k-1}, ..., u_2 = v)$ is a path from u to v in G. Notice in particular that P does not use the additional edge!

Question 2b. Let G be an undirected graph. Show that if G is minimally connected, then G is a tree.

Intuition: If you remove an edge from a cycle, you can still reach any vertex in the cycle from any other vertex in the cycle by going around the other way.

We'll start by showing that removing any edge from a cycle, you are still left with a connected graph.

Suppose $C = (u_1, u_2, ..., u_k = u_1)$ is a cycle, and let u_i, u_j be any two distinct vertices. Then there are two paths from u_i to u_j $(u_i, u_{i+1}, ..., u_j)$, and $(u_i, u_{i-1}, ..., u_j)$. Let $\{u_\ell, u_{\ell+1}\}$ be any edge in the cycle, and notice that it is in at most one of the paths. Thus, if it were to be removed, there would still be a path from u_i to u_j.

Now, let $G = (V, E)$ be minimally connected. We need to show that G is also acyclic. By contradiction, assume G has a cycle $C = (u = u_1, v = u_2, ..., u_k = u)$.

We claim that $G - \{u, v\}$ (G with the edge $\{u, v\}$ removed), is still connected (which is a contradiction, since G was supposed to be minimally connected).

Let x, y be any two distinct vertices in G. Since G is connected, there is a path $P = (x = x_1, ..., x_n = y)$ from x to y.

If this path doesn't use the edge $\{u, v\}$, then we're done since this path still exists in $G - \{u, v\}$.

Otherwise, the path includes the edge $\{u, v\}$, Let i and j be the indices where P 'enters' and 'leaves' the cycle, that is i is the smallest index such that $x_i \in C$, j is the largest index such that $x_j \in C$. By the earlier claim, there is still a path from x_i to x_j, furthermore, the path from x_i to x_j uses only vertices in C and hence doesn't intersect with the path from x to x_i and the path from x_j to y.

3 Induction

Question 3a. Show that for all $n \in \mathbb{N}$ with $n \geq 1$,

$$\left(1+\frac{1}{1}\right)\left(1+\frac{1}{2}\right)\left(1+\frac{1}{3}\right)...\left(1+\frac{1}{n}\right) = n+1$$

Let $P(n)$ be the predicate

$$\left(1+\frac{1}{1}\right)...\left(1+\frac{1}{n}\right) = n+1$$

By induction.

Base case. For $n = 1$, we have $(1+1/1) = 1+1$. Thus, the base case holds. **Inductive step.** Let $k \in \mathbb{N}$ be any natural number with $k \geq 1$, and assume $P(k)$. We'll show $P(k+1)$.

$$\begin{aligned}\left(1+\frac{1}{1}\right)...\left(1+\frac{1}{k}\right)\left(1+\frac{1}{k+1}\right) &= (k+1)\left(1+\frac{1}{k+1}\right)\\ &= (k+1)+\frac{k+1}{k+1}\\ &= (k+1)+1,\end{aligned}$$

where the first equality holds by the induction hypothesis. as required.

Question 3b. Show by induction $\forall n \in \mathbb{N}.(n^3 \leq 3^n)$

We first note that the claim holds for $n = 0, 1, 2$ since $0^3 = 0 \leq 3^0$, $1^3 = 1 \leq 3^1$, and $2^3 = 8 \leq 3^2 = 9$.

We will now show that $\forall n \in \mathbb{N}, n \geq 3.(n^3 \leq 3^n)$. By induction.

Base case. For $n = 3$, both the LHS and the RHS of the equation are 3^3 and thus equal. **Inductive step.** Let $k \in \mathbb{N}$, with $k \geq 3$ be any natural number at least 3. Suppose $k^3 \leq 3^k$. Then we have the following

$$\begin{aligned}
(k+1)^3 &= k^3 + 3k^2 + 3k + 1 \\
&\leq k^3 + k^3 + k^2 + k && (k \geq 3) \\
&\leq k^3 + k^3 + k^2 + k^2 && (k^2 \geq k) \\
&\leq k^3 + k^3 + 3k^2 && (3k^2 \geq 2k^2) \\
&\leq k^3 + k^3 + k^3 && (k \geq 3) \\
&= 3k^3 \\
&\leq 3 \cdot 3^k && \text{(IH)} \\
&= 3^{k+1}
\end{aligned}$$

which completes the induction.

Question 3c.

Your friend claims the following. **Claim:** $a^n = 1$ for all $n \in \mathbb{N}$, $a \in \mathbb{R}_{>0}$, and proposes the following proof by complete induction.

Proof. **Base case.** The base case holds since $a^0 = 1$ for any non-zero a.

Inductive step. Let $k \in \mathbb{N}$ be any natural number and assume for every $m \leq k$, and $a \in \mathbb{R}_{>0}$, $a^m = 1$. Then we have

$$a^{k+1} = a^{k+k-(k-1)} = \frac{a^k \cdot a^k}{a^{k-1}} = \frac{1 \cdot 1}{1} = 1,$$

which completes the induction. □

Find the flaw in the following argument, and explain the mistake.

You can't use a^{k-1} in the inductive step since when $k = 0$, $k - 1$ is not a natural number and hence is not included in the inductive hypothesis.

4 Round Robin

A round-robin tournament is a tournament in which every player plays a single game (rock-paper-scissors, tennis, chess, etc.) with every other player.

If V is a set of players, we encode the results of a tournament as a directed graph $G = (V, E)$, $E = \{(p, q) \in V \times V : p \text{ beats } q\}$

Question 4a. Let $G = (V, E)$ be a round-robin tournament. Here are two more definitions.

- A player p is a winner if for all other players q, either p beat q or p beat someone who beat q.
- A player p has a maximal number of wins if there is no other player q with strictly more wins than p.

Is a player with the maximal number of wins always a winner? Prove your answer!

Let p be a player with the maximal number of wins, k. By contradiction, suppose p is not a winner, that is, there is some q that p did not beat, and p did not beat anyone who beat q. Let's count the number of people q beat. q beat p, AND everyone p beat, thus q beats at least $k + 1$ people, which is a contradiction to the fact that p is a player with the maximal number of wins.

Question 4b. Let $G = (V, E)$ be any round-robin tournament encoded as a directed graph. Show that if $|V| \geq 3$, and G has some cycle, G has a cycle of length exactly 3. Hint: Try the Well Ordering Principle!

Let $P(n)$ be the predicate that is true if and only if for every round robin tournament $G = (V, E)$ with $|V| = n$ with a cycle has a cycle of length 3. We are trying to prove $\forall n \in \mathbb{N}, n \geq 3.(P(n))$

If $|V| = 3$, and G has a cycle, it must be of length 3. Thus, $P(3)$ holds.

By contradiction, assume the claim was false so that the set $A = \{n \in \mathbb{N} : n \geq 3 \land \neg P(n)\}$ is non-empty. By the well ordering principle, A has some minimal element m. Let $G = (V, E)$ be a round robin tournament where $|V| = m$, G has a cycle but no cycle of length 3. Let $(x_1, ..., x_k)$ be a cycle in G with $k \geq 4$. There are two cases.

- If $(x_3, x_1) \in E$, then there is a cycle of length 3, namely (x_1, x_2, x_3, x_1), which is a contradiction.
- Otherwise, $(x_1, x_3) \in E$. Let G' be the graph obtained by throwing out x_2. Then $(x_1, x_3, ..., x_k)$ is a cycle in G', but there is still no cycle of length 3 in G' (throwing out

a vertex does not create a new cycle). Vince G' has $m-1$ vertices, $m-1 \in A$, which contradicts the minimality of m in A.

5 Rubik's Inevitability

A Rubik's cube is a fun puzzle/toy. If you don't have a Rubik's cube, check out this simulator.

Assume the blue face of the cube is always facing you. A state of the cube describes the current position of all the stickers. For example, the solved state of the cube is the one where every face has just one color (namely, the one matching the center square of the face).

The cube has 6 faces. Let f_i be the operation of turning the ith face of the cube a quarter turn clockwise. Let $\mathrm{Turns} = \{f_1, ..., f_6\}$ be the set of turns one can perform on the Rubik's cube.

Question 5a. Let s be the solved state of the Rubik's cube. Define the set of possible Rubiks cube states, Rubiks, recursively or inductively.

Rubiks is the smallest set such that

- $s \in \mathrm{Rubiks}$
- If $x \in \mathrm{Rubiks}$, then $\forall i \in [6]$, $f_i(x) \in \mathrm{Rubiks}$.

Alternatively: Rubiks is the set generated from $\{s\}$ by Turns.

Question 5b. Model the task of solving a Rubik's cube in the minimum number of moves as a graph problem. Fully specify the vertex set, edge set, and edge weights (if any), and explain why a solution to the graph problem corresponds to solving the cube.

Let $s \in \mathrm{Rubiks}$ be the solved state of the cube. Define the following directed graph $G = (\mathrm{Rubiks}, E)$ where for any states $x, y \in \mathrm{Rubiks}$,

$$(x, y) \in E \iff \exists f \in \mathrm{Turns}.(f(x) = y).$$

I.e. there's an edge from x to y if and only if you can reach state y from state x with one turn.

A path in the G from $(x_1, ..., x_n)$ corresponds to a sequence of turns, that takes the cube from state x_1 to state x_n. Thus, the problem of solving a Rubik's cube from a scrambled state r in the fewest number of turns corresponds to the shortest path problem on input G, starting vertex r, and end vertex s.

Start with a solved Rubik's cube and consider any fixed sequence of turns. Now repeat that sequence of turns many times. **Eventually, somewhat magically, the Rubik's cube will return to the solved configuration!** In this problem, we will prove this fact.

Question 5c. Show that forall $f \in \text{Turns}$, f is an injective function from Rubiks to Rubiks.

Let $f_i \in \text{Turns}$ be the operation that turns face i a quarter turn clockwise. We'll show that f_i is injective. Note that from HW1, it suffices to find a left inverse for f_i. Let f_i' be the operation that turns face i a quarter turn counter-clockwise (equivalently, 3 quarter turns clockwise).

Let $x \in Rubiks$ be any state, $f_i'(f_i(x))$ starts at state x, turns the ith face a quarter turn clockwise and then a quarter turn counter-clockwise, clearly, we end up in the original, state x. Thus, f_i' is a left inverse of f_i and hence f_i is injective.

Question 5d. Prove by induction that for all natural numbers $n \geq 1$, if $g_1, g_2, ..., g_n \in \text{Turns}$, the composition $g_n \circ g_{n-1} \circ ... \circ g_1$ is injective.

First, we'll prove the following lemma.

Lemma. *The composition of two injective functions is injective.*

Proof of Lemma. Let $f : A \to B$, and $g : B \to C$, be injective functions. We'll show that $g \circ f$ is injective too. Suppose $g \circ f(x) = g \circ f(y)$, then by the injectivity of g, we have $f(x) = f(y)$, and by injectivity of f, we have $x = y$. Thus, $g \circ f$ is injective as required. ■

Let $P(n)$ be the predicate: For any sequence of k functions $g_1, ..., g_k \in \text{Turns}$, the composition $g_k \circ g_{k-1} \circ ... \circ g_1$ is injective. By induction.

Base case. For $n = 1$, let $g_1 \in \text{Turns}$. By the previous part, g_1 is injective, thus the base case holds.

Inductive step. Let $k \in \mathbb{N}, k \geq 1$ be any natural number at least 1, and assume $P(k)$. Let $g_1, ..., g_{k+1} \in \text{Turns}$ be a sequence of $k + 1$ functions. Let $h = g_k \circ g_{k-1} \circ ... \circ g_1$. By the induction hypothesis, since h is the composition of k functions in Turns, h is injective. Then $g_{k+1} \circ g_k \circ g_{k-1} \circ ... \circ g_1 = g_{k+1} \circ h$. Since $g_{k+1} \in \text{Turns}$ and is injective by the previous part, and h is injective, and the composition of two injective functions is injective (by the lemma above), $g_{k+1} \circ g_k \circ g_{k-1} \circ ... \circ g_1$ is injective as required.

Question 5e. Give an upper bound on $|\text{Rubiks}|$ i.e. find some k such that $|\text{Rubiks}| \leq k$

There are 6 colors on the Rubik's Cube, and $9 \times 6 = 54$ stickers. Thus, there are at most

$$6^{54}$$

states of the Rubik's Cube (each sticker can be one of 6 colors).[2]

Let g be an arbitrary sequence of turns. Denote

$$g^m = g \underbrace{\circ ... \circ}_{m \text{ times}} g$$

to be the function that applies g m times. Also, let $s \in \text{Rubiks}$ be the solved state of the cube.

Question 5f. Show that for some $m \in \mathbb{N}$ with $m \geq 1$, $g^m(s) = s$. That is, after m applications of the seqeunce of moves defined by g, we return to the solved state!

Hint: Consider the sequence $s_0, s_1, s_2, ...$ where $s_0 = s$, and $s_i = g^i(s)$.

Let g be any sequence of turns. Let k be the upper bound on Rubiks. Let $s = s_0$ be the solved state and $s_0, ..., s_k \in \text{Rubiks}$ be the sequence of states obtained by repeatedly applying g. I.e. for $i \in \mathbb{N}, i \geq 1$, $s_i = g^i(s)$.

Note that since there are $k+1$ states in the sequence, and $|\text{Rubiks}| \leq k$, by the pigeonhole principle, there is at least one state that appears at least twice in the sequence. Let $m \in \mathbb{N}$ with $m \leq k$ be the first index for which the sequence $s_0, ..., s_m$ has some state appearing twice. I.e. $s_0, ..., s_{m-1}$ are distinct and $s_m = s_i$ for some $i \in \mathbb{N}$, $i < m$.

We'll show that i must be 0. By contradiction, suppose $s_m = s_i$ for some $i > 0$. Then $g(s_{i-1}) = g(s_{m-1})$. Since $i < m$, and m is the first index for which there is a repeated state, $s_{i-1} \neq s_{m-1}$ - but this contradicts the fact that g is injective!

Thus, $g^m(s) = s$.

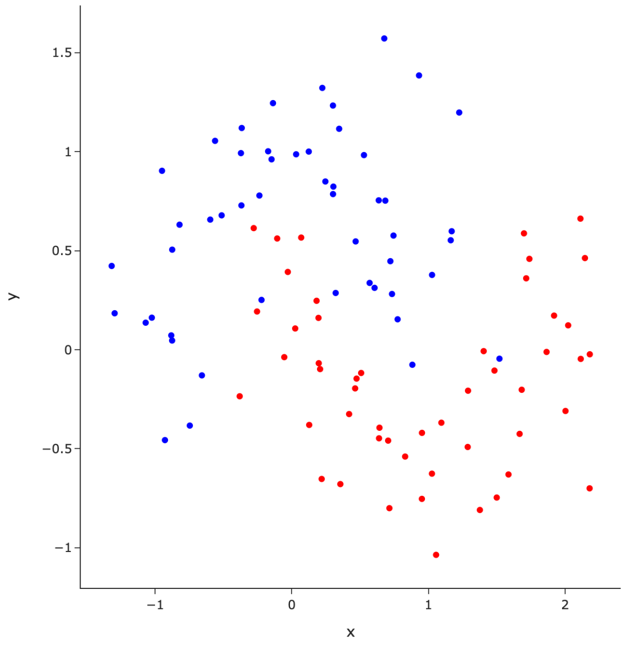
[2]Note that this is a gross overestimate since there are only 9 stickers of each color. If you tried to calculate the number of states exactly, you did more work than was required - I just asked for an upperbound!

6 A Neural Network Breakthrough?

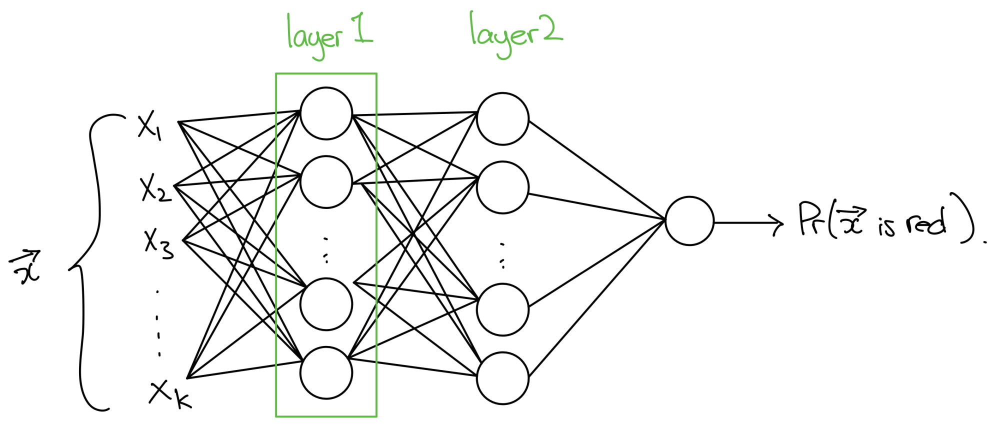
Your friend approaches you with a new idea, claiming it is possibly a breakthrough in neural networks for binary classification.

Binary classification is the problem where you're given some input and asked to classify the input as one of two classes. One example of binary classification is to predict whether or not the patient has pneumonia based on X-ray images. In this problem, we will focus on predicting whether a point is blue or red given the x and y coordinates.

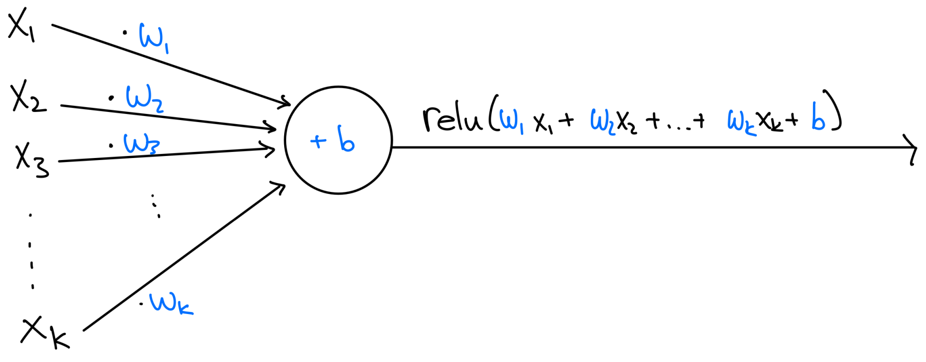
Our data looks like this:



A neural network looks like the following.



The circles are called neurons and compute as follows. Zooming on a single neuron:



Let k be the number of inputs for a particular neuron. Each neuron stores $k+1$ parameters, one weight, $w_i \in \mathbb{R}$, for each input, and one additional parameter called the bias, $b \in \mathbb{R}$. The output of the neuron is $\mathrm{relu}(w_1x_1+w_2x_2+\ldots+w_kx_k+b)$. In words, the neuron computes the weighted sum of the inputs, adds the bias term, and applies relu. $\mathrm{relu} : \mathbb{R} \to \mathbb{R}$ is function defined as follows

$$\mathrm{relu}(x) = \begin{cases} 0 & x \leq 0 \\ x & x > 0 \end{cases}$$

Training a neural network amounts to finding the 'best' setting of the weights and biases according to the data.

Neurons are then arranged into layers so that the first layer gets the inputs from the actual inputs, and subsequent layers get their inputs from the outputs of the previous layers! One reason neural networks are so powerful is that you can have many layers, each transforming the data in more complex ways. The size of a layer is the number of neurons in it.

The final layer contains a single neuron whose output is the probability that the input is red. Then, if the probability is ≥ 0.5, we'll predict the point is red and otherwise blue.

The final neuron is very slightly different from the other neurons. It still computes a weighted average of its inputs plus a bias. However, it applies a different function called σ ('sigmoid') instead of relu.

Here is your friend's idea:

Look carefully at what relu *does. If the input to relu is positive, we report the value, but if it is negative, relu just outputs 0. Isn't that wasting a lot of information? Shouldn't it be useful for later neurons to know how negative the input was? So here is my idea, instead of having a neuron output* $\mathrm{relu}(w_1x_1 + w_2x_2 + \ldots + w_kx_k + b)$*, have it output* $w_1x_1 + w_2x_2 + \ldots + w_kx_k + b$*.*

Question 6a. Test this theory out for yourself here.

First, run the model as is to check that it works.

Then, to test your friend's theory, remove the `activation="relu"` optional arguments in the definition of the model and try rerunning it. Try again with more and bigger layers.

How does the model do? Does your friend's idea work?

Question 6b. Unfortunately, your friend's idea doesn't work. Prove the following statement to show this:

If neurons compute in the suggested way, then any neural network with an arbitrary number of layers with arbitrary sizes is equivalent to some single output neuron (and thus, can't be all that powerful!)

First we'll make an important observation. Suppose you have a layer that takes k_{in} inputs and has size k_{out}. Let $\vec{x}$ be the input to this layer. Then if you write

$$W = \begin{bmatrix} \text{—}\vec{w}_1\text{—} \\ \text{—}\vec{w}_2\text{—} \\ \vdots \\ \text{—}\vec{w}_{k_{out}}\text{—} \end{bmatrix}, \vec{b} = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_{k_{out}} \end{bmatrix}$$

where $\vec{w}_i$, and b_i are the weights and the biases of the ith neuron in the layer, the output of the layer is $W\vec{x} + \vec{b}$ (by which I mean, the output of the ith neuron in the layer is the ith component of the vector $W\vec{x} + \vec{b}$).

We'll now show the following holds.

$\forall l \in \mathbb{N}$, any neural network with l layers (not including the output or input layers) is equivalent to some 0 layer neural network (which is just a single output neuron).

By induction on l.

Base case. The base case is true since a 0 layer neural network is equivalent to itself, which is a 0 layer neural network.

Inductive step. Let $l \in \mathbb{N}$ Suppose the claim is true for l-layer neural networks. We'll show that it is also true for $l + 1$ layer neural networks. Consider an arbitrary $l + 1$ layer neural network N, and look separately at the first layer and then last l layers.

Then the last l layers is a neural network with l layers. By the inductive hypothesis, it is equivalent to some single output neuron. Let $\vec{w_2}$ and b_2 be the weight vector and bias of this equivalent output neuron.

Let W, and $\vec{b}$ be the weights and biases of the first layer.

Then, on input x, N outputs $\sigma(\vec{w_2}^T(Wx + \vec{b}) + b_2)$. Using facts about matrix multiplication (namely distributivity, and associativity), this is equal to $\sigma((\vec{w_2}^T W)x + (\vec{w_2}^T\vec{b} + b_2))$, and this is computed by a single output neuron with weight vector $\vec{w_2}^T W$, and bias $\vec{w_2}^T\vec{b} + b_2$. This completes the induction.