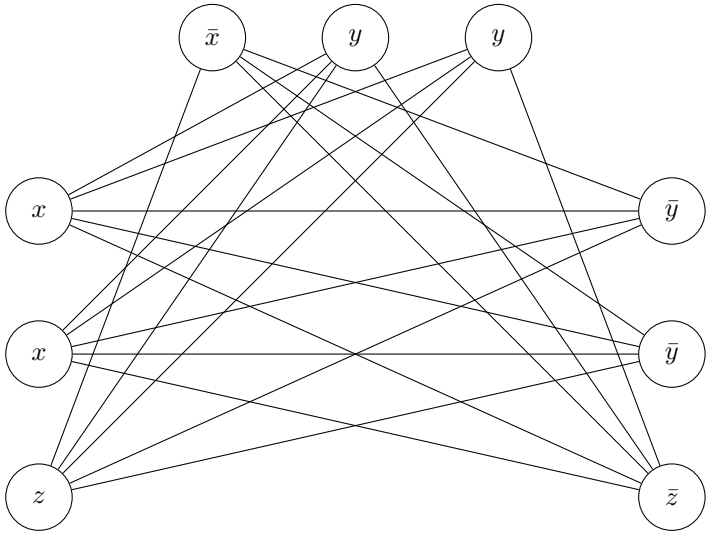
Homework 8

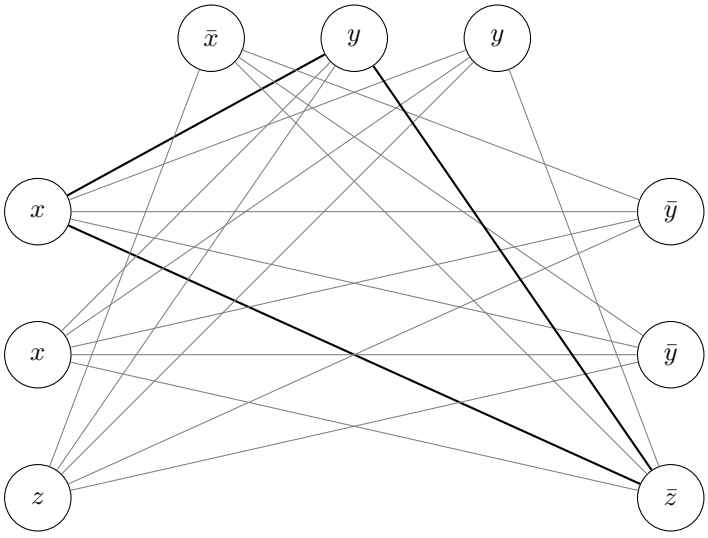
Walker Bagley

April 28, 2023

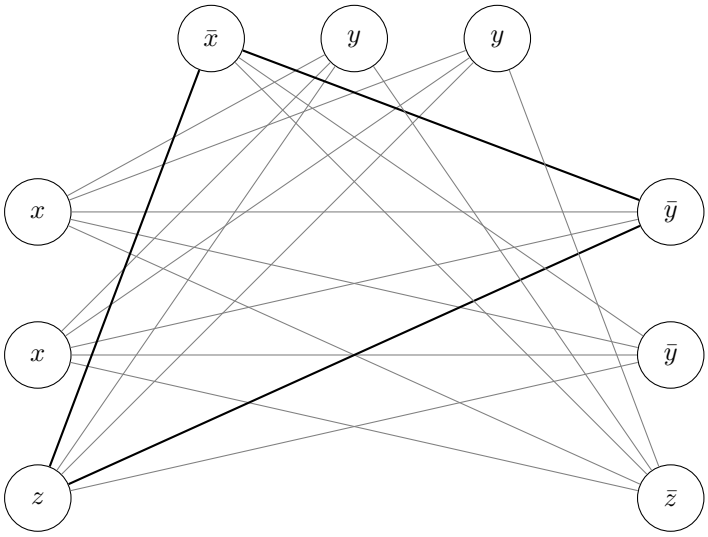
1. (a) $k = 3$ for $\phi = (x \vee x \vee z) \wedge (\bar{x} \vee y \vee y) \wedge (\bar{y} \vee \bar{y} \vee \bar{z})$



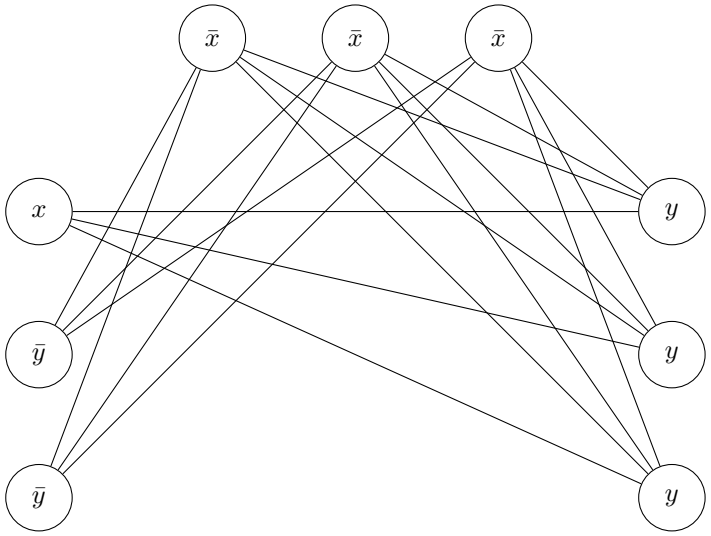
(b) i. Truth assignment: $x = 1, y = 1, z = 0$
Subsets: 4
Is a clique of V



ii. Truth assignment: $x = 0, y = 0, z = 1$
Subsets: 2
Is a clique of V



(c) $k = 3$ for $\phi = (x \vee \bar{y} \vee \bar{y}) \wedge (\bar{x} \vee \bar{x} \vee \bar{x}) \wedge (y \vee y \vee y)$



ϕ is not satisfiable since clause two requires $x = 0$ and clause three requires $y = 1$ in any solution to ϕ. However, we need $x = 1$ or $y = 0$ in clause one so we have a contradiction, ϕ cannot be satisfied. It is apparent from the graph V above that there are no cliques since there are no triangles connecting nodes on each of the three sides.

2. (a) $\{+\}$-DIGITS= $\{\langle t, S\rangle \mid (t, S, \{+\})$ is solvable $\}$

Proof.
Part 1: Show that $\{+\}$-DIGITS is in NP
If S contains n numbers, then we can check whether a subset C of S sums to t in $\mathcal{O}(n)$ time.

Part 2: Show that $\{+\}$-DIGITS is NP hard
By reduction from 3SAT, we have a mapping from formulas of ϕ to pairs $\langle t, S\rangle$ defined by:

i. Each number l in S has $n = v + c$ digits where v is the number of variables and c is the number of clauses
ii. Each variable and its negation set l_i to 1 where i is the enumeration of the variable, with the rest of $l_1 \ldots l_v$ digits being 0 (so if we have variables x, y, z, x and $\bar{x}$ start with 100, y and $\bar{y}$ start with 010, etc.)

iii. Each variable and negation assigns the number of times its literal appears in each clause i as l_{v+i}

iv. We have numbers in S for each literal, but also extras that help us add to t

v. For each clause i with w literals, we include $w-1$ l's in S where l is all 0's and $l_{v+i}=1$

vi. Then $t = 1^v 3^c$ where v, c are the respective number of variables and clauses

If the 3SAT formula is satisfiable, then there will be a subset of S that adds to t and if it is not satisfiable, then there will be digits representative of each clause that are unable to add to 3. From this construction, we can see how to reduce 3SAT to $\{+\}$-DIGITS and check its solvability in polynomial time. Further, we completed this reduction in class with the same construction for SUBSET-SUM. □

(b) $\{+,\times\}$-DIGITS= $\{\langle t, S\rangle \mid (t, S, \{+,\times\})$ is solvable $\}$

Proof.
Part 1: Show that $\{+,\times\}$-DIGITS is in NP
If S contains n numbers, then we can check whether a subset C of S can apply $+,\times$ to all elements of C in such a way that the end product is equal to t. Since there are n numbers, there are $n-1$ operations and 2 choices for each operation, so this check is performed in $\mathcal{O}(2^{n-1})$ time.

Part 2: Show that $\{+,\times\}$-DIGITS is NP hard
By reduction from $\{+\}$-DIGITS, we have a mapping from pairs $\langle t, S\rangle$ to $\langle t', S'\rangle$:

i. Set $t' = t * (t+1)$

ii. Then we construct $S' = \{s * (t+1) \mid s \in S\}$

So, if $\langle t, S\rangle$ is solvable, we have some $C = \{s_1 \dots s_i\}$ where $\Sigma C = t$ and thus some subset $C' = \{s'_1 \dots s'_i\}$ where $s'_n = s_n * (t+1)$, so $\Sigma C' = (t+1) * \Sigma C = t * (t+1) = t'$. Conversely, if we have some mapping of $\langle t, S\rangle$ to $\langle t', S'\rangle$ with some subset C' that solves $\langle t', S'\rangle$, then we know that $t' = t * (t+1)$ and $S = \{\frac{s'}{t+1} | s' \in S'\}$, which is obviously solvable in $\{+\}$-DIGITS. □

(c) $\{+,-\}$-DIGITS= $\{\langle t, S\rangle \mid (t, S, \{+,-\})$ is solvable $\}$

Proof.
Part 1: Show that $\{+,-\}$-DIGITS is in NP
If S contains n numbers, then we can check whether a subset C of S can apply $+,-$ to all elements of C in such a way that the end sum is equal to t. Since there are n numbers, there are $n-1$ operations and 2 choices for each operation, so this check is performed in $\mathcal{O}(2^{n-1})$ time.

Part 2: Show that $\{+,-\}$-DIGITS is NP hard
By reduction from 3SAT, we use the same construction as SUBSET-SUM and $\{+\}$-DIGITS. We know that this construction works with addition, but we must show that subtraction does not impact the satisfiability of the mapping. Consider some unsatisfiable ϕ and its mapping to an input $\langle t, S\rangle$. We know that no subset C of S can purely add to t. Then for any subset that adds to a sum greater than t, we show that we cannot subtract any remaining numbers in S to reach t. Breaking the sum into two parts for the variables and clauses, if $\Sigma C > t$ then we have more than one of each variable, or more than three of some clause. In case 1, if both the positive and negative literals for a variable were used, then neither remains in S to subtract from the total, so we cannot reach t. In case 2, if one of the clauses adds to more than 3, then there are not enough slack numbers to subtract from that clause since they were already used to add to the particular clause. □

3. *Proof.*
Part 1: Show that the neural network is in NP
If there are l inputs and m hidden units in the neural network, then given arbitrary values for u, a, v, b we can check if there exist a sequence of inputs that makes $y = 1$. Consider a sequence of inputs x, then it takes $l * m$ operations to generate the hidden units and then another m operations to generate y, so we can check whether a given sequence makes $y = 1$ in $\mathcal{O}(m * (l+1)) \in \mathcal{O}(l^2)$ time.

Part 2: Show that the neural network is NP hard
By reduction from 3SAT, we have a mapping from formulae ϕ to inputs $\{x_1 \dots x_l\}$:

(a) Let each unique variable be an input of the neural network, enumerated $x_1 \dots x_l$ for the l variables. Each variable will be set to a 1 or a 0 for its truth value

(b) Let m be the number of clauses in ϕ, where h_1 is the first clause, h_2 the second, etc.

(c) Assign u_{ij} for variable i and clause j as follows:

$$u_{ij} = \begin{cases} 1 & \text{if there is at least one "positive" } i \text{ literal [ex: } (i \vee x_1 \vee x_2) \text{ or } (i \vee \bar{i} \vee x_1)] \\ 0 & \text{if there are no } i \text{ literals [ex: } (x_1 \vee x_2 \vee x_3)] \\ -1 & \text{if all } i \text{ literals are "negative" [ex: } (\bar{i} \vee x_1 \vee x_2) \text{ or } (\bar{i} \vee \bar{i} \vee x_1)] i \end{cases}$$

(d) Assign a_j to be the number of unique negated variables in clause j (don't double count $\bar{i}$)

(e) Assign $v_j = 1$ for each clause j since we want the construction of y to be true iff all our clauses are true

(f) Assign $b = -(m-1)$ where we recall that m is the number of clauses in ϕ

We know that if some ϕ in 3SAT is satisfiable, that each clause is true for some input of the variables in ϕ. Then with this construction, we create an intermediary node h for each clause in ϕ that evaluates to 1 if the clause is true on a given input and 0 if the clause is false. By utilizing the same input for both positive and negative literals of the same variable, we ensure no contradiction between literal values in each "clause" node. Further, the assignment of u_{ij} keeps track of the true literals for a node while a_j corrects for any discrepancies caused by negated variables. So we can say that if ϕ is satisfiable, then there is some input of variables that will yield $y = 1$ in our neural network. Conversely, if there is a sequence of inputs that satisfy our neural network, then we can reverse the above construction, using the variable configuration as a satisfiable solution of ϕ. □