Homework 08

Aaron Wang

April 29 2025

1. Define a *neural network* with l inputs and m hidden units to be a function that takes a sequence of inputs $x_1, \ldots, x_l \in \{0, 1\}$ and computes

$$h_j = H\Big(\sum_{i=1}^{l} x_i u_{i,j} + a_j\Big) \qquad\qquad j = 1, \ldots, m$$

$$y = H\Big(\sum_{j=1}^{m} h_j v_j + b\Big)$$

where

$$H(x) = \begin{cases} 1 & \text{if } x > 0 \\ 0 & \text{otherwise} \end{cases}$$

The weights $u_{i,j}$, a_j , v_j , and b can be any rational numbers, and they don't depend on $x_1, \ldots, x_l$. Let's say that the size of the neural network is the number of units ($n = l + m + 1$). Prove that it is NP-complete to decide, given a neural network with l inputs, whether there is any sequence of inputs $x_1, \ldots, x_l$ that makes $y = 1$.

Let NEURAL= $\{\langle N \rangle \mid N$ is a *neural network* described above and for some sequence of inputs $x_1, ..., x_l$ that makes $y = 1\}$.

To prove NEURAL is NP-complete, we must show it is both NP and NP-hard.

Let V be a verifier for NEURAL.

V = "On input $\langle N, X \rangle$ where N is a neural network and X contains all the inputs $x_1, ..., x_l$:

1. $\forall j$ solve h_j.
2. Solve y.
3. If $y = 1$, *accept*; otherwise, *reject*.

Step 1 is $\mathcal{O}(l * m)$ and step 2 is $\mathcal{O}(m)$. Consequently, V is definitely $\mathcal{O}(n^2)$. Since we have a deterministic polynomial time verifier, we know NEURAL is NP.

Here are the details of a reduction from 3SAT to NEURAL that operates in polynomial time.

1. For each variable x_i:
   - Create nodes x_i and x_{-i}.
   - Create node $h_{x_i} = H(x_i + x_{-i})$
   - Create node $h_{x_{-i}} = H(-x_i - x_{-i} + 2)$
2. For each clause j: create h_j s.t.
   - if x_i is in the clause, $u_{i,j} = 1$
   - if $\overline{x_i}$ is in the clause, $u_{-i,j} = -1$
   - otherwise $u_{i,j} = 0$
3. $v_j = 0$ or $y = H(\sum_{j=1}^{m} h_j - m + 1)$

Following this reduction, we have mapped each boolean expression to a neural network that is satisfiable iff the boolean expression is. Step 1 initializes nodes for x_i and $\overline{x_i}$ and ensures that x_i and $\overline{x_i}$ must be opposites. Step 2 translates all the clauses into the neural network. Step 3 essentially ands every clause together. Step 1 is $\mathcal{O}(n)$, 2 is $\mathcal{O}(n^2)$, and 3 is $\mathcal{O}(n)$. This is clearly a $\mathcal{O}(n^2)$ reduction. Thus, NEURAL is NP-complete.

2. In the game of Digits, you are given

   - A target number t (e.g., 56)
   - A multiset S of source numbers (e.g. $\{2, 3, 4, 5, 10, 25\}$)
   - A set of $\mathcal{O}$ of operations ($\{+, -, \times, \div\}$)

   and the goal is to write an expression involving the source numbers, operations, and parentheses that evaluates to the target number. You don't have to use every number, but each number can only be used as many times as it occurs in S. If this possible, we say that $(t, S, \mathcal{O})$ is solvable. For example, $(56, \{2, 3, 4, 5, 10, 25\, \{+, -, \times, \div\})$ is solvable because

   $$2 \times 4 \times (10 - 3) = 56$$

   Prove that $\{+, \times\}$-DIGITS is NP-complete:

   $$\{+, \times\}\text{-DIGITS} = \{\langle t, S\rangle | (t, S, \{+, \times\}) \text{ is solvable}\}.$$

   Assume that the size of $\langle t, S\rangle$ is the total number of bits in t and S.

   To prove $\{+, \times\}$-DIGITS is NP-complete, we must show it is both NP and NP-hard.

   Let V be a verifier for $\{+, \times\}$-DIGITS.

   V = "On input $\langle t, S, c\rangle$ where t and S are as described above and c is the certificate, a sequence (order matters) of 3-tuples representing (n_1, n_2, operation):

   1. For each tuple in c
      i. if n_1 and n_2 not both in S, *reject*
      ii. remove n_1 and n_2 from S.
      iii. n is the result from performing the operation[1] on n_1 and n_2.
      iv. Add n into S
   2. If $t \in S$, *accept*; otherwise *reject*

   V is $\mathcal{O}(n)$ where $n = |S|$ since $|c| \leq |S|$. Since we have a deterministic polynomial time verifier, we know $\{+, \times\}$-DIGITS is NP.

   Here are the details of a reduction from SUBSET-SUM[2] to $\{+, \times\}$-DIGITS that operates in polynomial time. We will map from $\langle t, S\rangle$, an instance of SUBSET-SUM to $\langle t', S'\rangle$ an instance of $\{+, \times\}$-DIGITS .

   1. Let $t' = t \times (t + 1)$
   2. $S' = \{n \times (t + 1) | n \in S\}$

   Following this reduction, we have mapped an instance of SUBSET-SUM to a $\{+, \times\}$-DIGITS that is satisfiable iff the the original problem is. We multiplied each number by $t+1$ essentially to render multiplication useless. Observe, that there is no way to lower a number. In this new construction, if we ever use multiplication, the new number becomes useless because it is $> t$. Thus, we would only ever be able to use addition as imposed by subset sum. This is clearly $\mathcal{O}(n)$ reduction. Thus $\{+, \times\}$-DIGITS is NP-complete.

[1] If the operation is not + or × obviously *reject*
[2] SUBSET-SUM where all numbers are in the positive naturals.

3. You are at the entrance to a room whose floor is made entirely of trapdoors, and underneath the floor is a pool of lava. Each trapdoor is either open or closed.

   At the entrance to the room is a control panel that has buttons labeled with symbols. For any symbol σ, if you push the button labeled σ, then all the trapdoors labeled σ that are closed become open, and all the trapdoors labeled σ that are open become closed.

   In general, a room can be any size rectangle, with an entrance and exit anywhere around the perimeter, any initial pattern of open/closed trapdoors, and any number of labels/buttons.

   Prove that it is NP-complete to decide, given a room as defined above, whether there is a subset of buttons that you can push to make a safe path from the entrance to the exit.

   Let SAFEROOM = $\{\langle R\rangle \mid R$ is a room that can have a safe path$\}$

   To prove SAFEROOM is NP-complete, we must show it is both NP and NP-hard.

   Let V be a verifier for SAFEROOM.

   V = "On input $\langle R, c\rangle$ where R is as described above and c is the certificate, a set of toggled buttons:

   1. For each tile in the grid $T_{i,j}$, decide if it is closed or open by...
      i. If $R_{i,j}$ has letter σ and $\sigma \in c$, $T_{i,j}$ is the inverse of $R_{i,j}$'s initial state.
      ii. $T_{i,j}$ is $R_{i,j}$'s initial state otherwise.
   2. If the entrance tile is open, *reject*
   3. Mark every tile that is adjacent to a closed, marked tile and is also closed.
   4. Repeat step 3 until no new tiles are marked.
   5. If the exit tile is marked, *accept*; otherwise *reject*.

   This essentially figures out of each tile is open or closed based on the initial state and whether it has been toggled. It then runs a bfs and accepts if there is a path. V is polynomial time bounded by the size of the rectangle. Since we have a deterministic polynomial time verifier, we know SAFEROOM is NP.

   Here are the details of a reduction from 3SAT to SAFEROOM that operates in polynomial time. We will map from a boolean expression to $\langle R\rangle$ an instance of SAFEROOM .

   1. Let R be a room that is 3 rows by $2m + 1$ columns where $m =$ the number of clauses.
   2. Set the start tile to $T_{1,1}$ where we are using a 1-based indexing.
   3. For every tile in an odd column, set the initial state to closed and do not attach a letter to it.
   4. For every clause c_j that looks like $a \vee b \vee c$...
      i. $T_{1,2j}$ is open if a is a negated variable; it is closed otherwise. Additionally, $T_{1,2j}$ has symbol of the variable.
      ii. Same for $T_{2,2j}$ and b.
      iii. Same for $T_{3,2j}$ and c.
   5. Set the exit tile to $T_{1,2j+1}$.

   Following this reduction, we have mapped an instance of 3SAT to SAFEROOM that is satisfiable iff the the original problem is. Essentially, we have created a buffer between every clause in which you can choose which "gate" to go through next. We then transform the clauses into a wall of gates, which can all be crossed iff the corresponding clause was satisfiable. This is clearly $\mathcal{O}(n)$ reduction where n is the number of clauses. Thus SAFEROOM is NP-complete.