CS Theory (Fall '25) Assigned: Nov 18, 2025

Homework 5

Instructors: *William Pires and Toniann Pitassi* Due: Nov 30, 2025 at 11am

Collaboration: Collaboration with other students in the class homework is allowed. However, you must write up solutions by yourself and understand everything that you hand in. You may also consult other reference materials, but you may not seek out answers from other sources or use AI tools. **You are required to list who you collaborated with on each problem, including any TAs or students you discussed the problems with in office hours. Also list any reference materials consulted other than the lectures and textbook for our class.** See the course webpage for more details.

Formatting: Write the solution to each part of each problem on a separate page. Be sure to correctly indicate which page each problem appears on in GradeScope. Please do not write your name on any page of the submission; we are using anonymous grading in GradeScope. If we can't find your solution to any problems because it was not properly tagged with the page, or if handwriting is not legible, you will receive 20 percent on these problems.

Grading: There are 60 total possible points. For all problems you have the option of answering "Don't know" on any parts of the question, and you will receive 20 percent of the total marks for those parts.

0 Exercises

Here are some recommended exercises. You should not turn in your solutions for these, and we will not grade them.

(a) Sipser, 3rd edition, problem 7.9

(b) Sipser, 3rd edition, problem 7.22

(c) Sipser 3rd edition, problem 7.23 (solution in book)

(d) Sipser, 3rd edition, problem 7.40 (solution in book)

(e) Let Factor be the function that takes as input a natural number x in decimal notation, and outputs the prime factorization of x, also in decimal. For example, if the input is 13, then the output would be 13, and if the input is 12, the output would be 2,2,3. (You can output the prime factors in any order.) Prove that if P = NP then there is a polynomial-time algorithm for Factor. Note that NP is a class of *languages* (or decision problems), so factoring is not in NP. Therefore, you should give an algorithm for Factor, assuming that every language in NP is solvable in polynomial-time.

(f) Try to prove that the following languages are NP-complete. (Sipser, Section 7.5 covers all of these in detail but try to do them yourself first for practice.) Clique, Independent Set, HamPath, Vertex-Cover, Subset-Sum.

Please read the instructions at the end of this HW carefully before doing the problems. In particular, you may only use the 3 NP-complete problems defined there in your reductions.

1 Neural Networks (20 points)

A simple 3-layer feed-forward neural network is described as follows (see the accompanying figure below):

- The input layer consists of n input nodes, denoted $x_1, \ldots, x_n$;
- The middle layer consists of m middle (hidden) units denoted $y_1, \ldots, y_m$. Each hidden unit computes a weighted threshold function of the inputs. Specifically for each hidden unit y_j, there is a weight vector $\overline{w}_j = w_{j,1}, \ldots, w_{j,n}$, $w_{j,i} \in \{-1, 0, 1\}$ and a threshold value $t_j \in \mathbb{Z}$ and the activation of y_i is 1 if $\sum_{i=1}^{n} w_{j,i} x_i \geq t_j$ and 0 otherwise.
- The output layer consists of one output node denoted by f. It takes as input the hidden layer activation values and has a weight vector $\overline{v} = v_1, \ldots, v_m$, $v_k \in \{-1, 0, 1\}$ and a threshold value $t \in \mathbb{Z}$ and outputs 1 if $\sum_{k=1}^{m} v_k y_k \geq t$ and 0 otherwise.

The input to N-NET consists of: (i) the first layer weight vectors $w_{j,i} \in \{-1, 0, 1\}$, $1 \leq j \leq m$, $1 \leq i \leq n$, and the associated thresholds $t_1, \ldots, t_m \in \mathbb{Z}$; (ii) the second layer weight vectors $v_1, \ldots, v_m \in \{-1, 0, 1\}$, and threshold $t \in \mathbb{Z}$. The threshold values $t_1, \ldots, t_m$ and t are all given in binary representation. The input is in the language if and only if there exists an assignment $\alpha \in \{-1, 1\}^n$ to the input variables $x_1, \ldots, x_n$ such that the output node evaluates to 1.

(a) Prove that N-NET is in NP. Note that in your proof of correctness, you will need to discuss the length of the input as a function of n, m and the threshold values.

(b) Prove that N-NET is NP-hard, and therefore by (a) and (b) it is NP-complete.

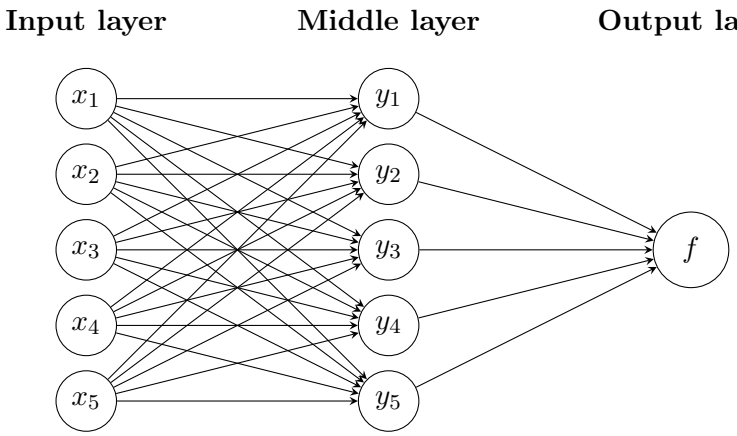

Figure 1: A 3–layer N-NET network with $n = 5$ and $m = 5$.

2 Fun with Math (25 points)

In the game of ChangePro, the input is a target number t, and a multiset S of coins where each coin has an associated positive integer value (e.g., $S = \{2, 3, 4, 4, 5, 7, 7, 10, 25\}$). The goal is to write an expression using the source numbers in S and the operations + and $\times$ that evaluates to the target number, t. The expression doesn't have to use all source numbers, but can only use each number as many times as it appears.

(a) Give a reasonable representation of the input to ChangePro as a string x over some finite alphabet. Specify your alphabet and encoding of the input. What is the length of x in terms of t and the numbers in S ?

(b) Using your representation, prove that ChangePro is in NP.

(c) Prove that ChangePro is NP-hard, and therefore by (b) and (c) it is NP-complete.

3 NP-Completeness (15 points)

Let L be the language consisting of all strings $w \in \{0, 1\}^*$ such that w contains at least 3 consecutive 1's. Prove that L is NP-complete if and only if P = NP.

Instructions on level of detail required for algorithms, reductions and proofs You do not need to discuss the mechanics of an actual TM (tapes, states, the transition function). See chapter 7 of the book for examples. Make sure your algorithm is clear and well defined (e.g., what is the input to the program, what are the main subroutines/loops/steps; specify the data structure(s) being used (variables, arrays, etc). You can give a few sentences to describe the ideas behind your algorithm, **but we expect your algorithm to be properly written pseudocode (with lines of codes, clear indentation etc...). Do not give us a big paragraph of text as your algorithm.**

We highly recommend that you number the lines of your algorithm to be able to refer to them easily.

Some clarifications:

Unless instructed, you should think of every numbers as being encoded in binary.

For Problems 1 and 2, when proving $L \in$ NP. You can either give an NTM or a verifier.

a) You have to prove your NTM / Verifier is running in polynomial time (use asymptotic notation).

b) You have to prove that if $x \in L$ your NTM accepts in some computation path, or that there is a certificate that makes your Verifier accepts. You should explain what the computation path / certificate is.

c) You have to prove that if $x \notin L$ your NTM rejects all computation paths, or that there are no certificates that make your Verifier accept.

For Problems 1 and 2, when proving L is NP-hard.

a) Give a polytime mapping reduction f from A to L where A is NP-hard (see below for which problems A you are allowed to use). Give this reduction as an algorithm in pseudo-code.

b) Prove that f can be computed in polynomial time.

c) You have to prove that if $x \in A$, then $f(x) \in L$.

d) You have to prove that if $x \notin A$, then $f(x) \notin L$.

NP-Hard problems you're allowed to use on the HW. You are only allowed to use the following NP-hard problems. Do not use other problems, even if we mentioned them in class or discussed in the book.

- In the following, S is a multiset of k positive integers[1], given in binary.

  SUBSET-SUM $:= \{\langle S, t\rangle \mid S = \{x_1, \ldots, x_k\},$ and for some $\{y_1, \ldots, y_l\} \subseteq \{x_1, \ldots, x_k\},$ we have $\sum y_i = t\}$

- CLIQUE $= \{\langle G, k\rangle \mid$ G is an undirected graph with a k-clique $\}$.

- 3-SAT $= \{\langle \Phi\rangle \mid$ Φ satisfiable Boolean formula with at most 3 literals per clause$\}$.

[1]We say multiset (and not set) because the same number can appear multiple times in S, for instance you can have $S = \{2, 4, 4, 7, 7\}$