Machine Learning

Recap

CS51 – Spring 2026

Final exam

- Section I: Wednesday May 13th, at 2:00-5:00pm (in our classroom)
- Section II: Friday May 15th, at 2:00-5:00pm (in our classroom)
- Can bring **two** double-sided sheets of hand-written notes
- Cumulative for the entire semester
- Go over all slides and accompanying code/circuits/automata.
- Try to solve the practice time problems before you see the solution.
- Make a pass over all assignments
- If you still have questions, come to special office hours (see slack for announcements)
- Use the following as a (non-exhaustive) guide

History and Ethics

- Similar to checkpoint 1

Boolean Algebra

- NOT, AND, OR, XOR operators
- De Morgan's laws
- Distributive law
- DNFs and CNFs
- Minterms and maxterms
- Boolean Algebra in Python

Numeral Systems

- Decimal numbers
- Numbers in other bases, particularly binary and hexadecimal
- Converting between bases, particularly to/from decimal and binary to/from hexadecimal

Binary Arithmetic and Representing Information

- Addition in other bases, particularly binary and hexadecimal
- Overflow
- Unsigned, signed numbers (magnitude and two's complement representation)
- Equivalence across systems
- Sign extension
- Shifting: logical and arithmetic right, left shifting
- Bitwise operators

Combinational circuits

- Gates and truth tables for NOT, AND, OR, XOR, NAND, NOR
- minterm expansion to find DNF
- Karnaugh maps (see relevant assignment)
- Half- and full-adders
- Decoders
- 7-segment displays

Sequential circuits

- Memory units and calculating the size of memory
- Multiplexer
- SR-latches

CS51 machine

- Architecture (how many registers, their purpose, etc.)
- instructions:
- CS51 machine programs
- Loops and conditionals
- Functions
- Decoding instructions (instructions to binary)

Note: You will not be asked to *write* any CS51 assembly

Introduction to Proofs

- Atomic and compound propositions
- Logical connectives including if then and if and only if
- Precedence
- Tautologies
- Satisfiability
- Logical equivalence

Loop invariants

- Know how to state pre- and post-conditions
- How to write a loop invariant
- How to prove correctness of an iterative algorithm:
  - Initialization
  - Maintenance
  - Postcondition
  - Termination

Proof by induction

- Revisit summation notation
- Know how to apply template for proof by induction

Proof by strong induction

- Understand the difference between weak and strong induction
- Floor, ceiling, modulo, prime, composite, divisible
- Apply the template for strong induction

Recursion

- Practice with writing recursive functions on integers, lists, strings

Turtle graphics

- Know basic commands and how they manipulate the turtle

Sequences and dictionaries

- Strings, lists, tuples, ranges
- Indexing
- Slicing
- Iteration
- Membership
- Immutable and mutable
- Methods vs functions
- Aliasing
- Call stack
- Dictionaries

Files and errors

- How to open a file to read it line by line or read all of its lines
- How to write in a file
- Structure of dealing with exceptions

Sorting Algorithms

- Bubble-,selection-, insertion sort
- Know how to run them and their basic properties

Asymptotic Running Analysis

- Big O notation
- Common classes of runtime complexities and what they mean in practice when we double the input size of the problem we try to solve
- Ordering classes of runtime complexities from best to worst
- Understanding how to count comparisons in different sorting algorithms to identify the big O for worst, average, and best case performance

Higher-order functions

- Understand that functions are objects and can be passed as parameters, be returned from functions, and created on the fly
- Look into higher_order_functions.py and practice calling its different functions
- Understand how the built-in map and filter work
- Anonymous functions using lambda
- sorted function

DFAs

- Understand that DFAs are finite automata that accept or reject words built out of a specific alphabet and which belong to a specific language
- You need a starting state but can have more than 1 accept state
- For every state, you need to be explicit about what transitions need to happen for each character of the alphabet
- Regular expressions using concatenation, Kleene star, and union

NFAs

- Similar to DFAs but for a state we could have no transition for a character (empty set), one or even more
- There is a way to translate an NFA to DFA
- Regular languages can be described by a DFA/NFA

Turing machines

- More powerful than DFA and NFAs
- Be comfortable with transitions

Machine learning

- Supervised (classification, regression, ranking), unsupervised (clustering), reinforcement learning
- Biological idea behind artificial neural networks
- Neuron/perceptron as a unit that receives inputs with certain weights and gets activated/excited/stimulated if the weighted sum is above threshold

Perceptron training

- Add one extra 1-input, assign random weights to the n+1 inputs, threshold is set at 0.
- Fix learning rate between 0 and 1
- For each training point, if the prediction was wrong, adjust the non-zero input weights by delta.