CMU ONLINE
Carnegie Mellon University

Syllabus Overview Sample: Computational Foundations for Machine Learning

Program: Machine Learning & Data Science Certificate Program

Faculty: William Cocke, Researcher and Teaching Assistant of Language Technologies and Human- Computer Interaction, School of Computer Science

Course Number: 11961 **Units:** 6

Zoom Class: Weekday Evening, ET

Course Description

This course provides a place for students to practice the necessary computational background for further study in machine learning. Topics covered include computational complexity, analysis of algorithms, proof techniques, optimization, dynamic programming, recursion, and data structures. The course assumes some background in each of the above, but will review and give practice in each. (It does not provide from-scratch coverage of all of the above, which would be impossible in a course of this length.) No coding is required, but students will be expected to read and analyze code in Python for assignments and C for the textbook.

Course Learning Objectives

By the end of the course, students should be able to apply the computational topics discussed above in order to understand and analyze common machine learning topics.

Faculty Introduction



William Cocke is a researcher and teaching assistant at the Language Technologies and Human- Computer Interaction, School of Computer Science.

Course Topics

Topic	Learning Objectives
Intro to the Course + Iteration and Induction	• Explain what a simple Python program does. • Define the base case, inductive hypothesis, and inductive step of a proof by induction. • Apply induction to prove identities about summation. • Recognize when to use induction to prove a statement.
Recursion	• Explain how to define a given set recursively. • Prove properties of a program using induction. • Utilize recursive programs to solve various problems. • Compare recursion and iteration as tools for programming.
Computational Complexity	• Define big-O notation. • Explain why we use big-O analysis to talk about efficiency in programs. • Identity the big-O runtime of a simple algorithm or program. • Compare common big-O classes to one another, for example, polynomials, logarithms, and exponentials .
Trees, Binary Trees, and Binary Search Trees	• Define trees, binary trees, and binary search trees. . • Explain how to use a binary search tree to locate an item in a data set. • Identify the elements of a tree and how they relate to each other. • Compare and Contrast Different transversal orderings: preorder, postorder, inorder. • Compute transversals of a tree and binary tree.
Lists and Dynamic Programming	• Define a list as an abstract data structure. • Compare and Contrast the array and linked list implementations of a list. • Calculate the computational complexity of different list operations depending on how a list is implemented. • Compute solutions to recursive problems using dynamic programming.
Data Structures	• Define a graph using a set of edges and a set of nodes. • Compare and contrast the adjacency matrix and adjacency list representations of a graph. • Calculate a Depth-First Search traversal of a graph from a given start node. • Calculate a Breadth-First Search traversal of a graph from a given start node. • Use Dijkstra's algorithm to find the shortest path between two nodes on a graph

	● Explain some of the constraints of Dijkstra's algorithm. ● Compare and contrast dense and sparse vectors in terms of computational costs associated with common operations
Message Passing on a Graph and Final Assignment	● Explain how the representation of a graph is not unique. ● Update the representations at a node in a GNN using a given aggregation function.

NOTE: This is a sample designed to provide an overview of topics covered in the course based on previous course syllabi. All students will be provided an updated and final syllabus at the start of the course. Faculty, content, and schedule are all subject to change.