Syllabus (Fall 2024)

Dr. Sunil Shende

Welcome to the Fall 2024 offering of the Big Data Algorithms course! The course is cross-listed with course catalog numbers 50:198:462 (for the undergraduate section) and 56:198:562 (for the graduate section).

> **!** Important
>
> Please bookmark the URL for the Student Resource List. This link is also available from the left menu on the Canvas site for the course.

Course Information

Instructor	Sunil Shende
Schedule	W from 6pm to 8:50pm
Classroom	CS-109 Cooper Classroom Building
Office	BSB 308
Office Hours	MW from 10am to 11am (in person); Zoom (TBA)
Email	shende AT camden DOT rutgers DOT edu
Telephone	**5-6122** (on campus)

Learning Goals and Outcomes

Data mining and machine learning are now ubiquitous in Computer Science and Data Science. Our approach in this course is to study the **fundamentals** and this involves both theory *and* implementation.

The course is designed to achieve two primary learning goals:

1. To understand and appreciate some of the most important concepts behind algorithms for big data. This includes (but is not limited to) elements of algorithm analysis and correctness, probabilistic reasoning, metrics for data *similarity*, linear algebra for data science, and other theoretical foundations of learning from data.

2. To implement and use relevant big data algorithms with Python libraries, with a focus on *avoiding* as much as possible, the indiscriminate use of APIs (Application Programming Interfaces) for popular data mining and machine learning related Python libraries and frameworks.

By way of outcomes, students are expected to gain theoretical fluency with:

- basic probability concepts and inequalities
- distance and similarity measures for data
- hash functions, and in particular, their use with locality-sensitive hashing and probabilistic data structures for data streams
- linear algebra fundamentals as applied to big data, such as vectors and matrices, eigenvalues, matrix decompositions and factorizations
- basic learning theory, including perceptrons, gradient descent, and unsupervised learning

In addition, students will be exposed to multiple Python libraries for data science geared towards big data analysis and visualization. Becoming proficient users of these libraries is an expected outcome.

> **Note**
>
> Students will be provided with **free resource credits** that can used to work with **GCP (the Google Cloud Platform)** and **free Quiklab credits** to learn how to use specific compute-storage components of GCP. We will use GCP as needed in the course (especially for the term project).

Textbooks

Selected content from the following freely available textbooks will be used during the course.

- **Mining of Massive Datasets** by Anand Rajaraman, Jure Leskovec and Jeffrey D. Ullman, **3nd edition**. You can download the book's contents free of charge from the book's website courtesy of the publisher; however, I encourage you to buy a hardcopy of this excellent book as a reference. This will be one of our primary textbook and will be referred to as **MMDS** in the rest of the syllabus.

> **Note**
>
> There are a few changes from the 2nd to the 3rd edition, namely the addition of some material on Spark and an entire chapter on *Deep Learning*
> The site also provides access to all the videos of lectures from the Stanford Coursera course based on the book. I may assign some of these videos as preparation for specific classes.

- **Foundations of Data Science** by Avrim Blum, John Hopcroft and Ravindran Kannan, 1st Edition, Cambridge University Press, 2020. The book is available for free download/online viewing from the Rutgers Libraries website. We will refer to this textbook as **FDS**.

Software

1. For coding purposes, we will use **Python 3.12+** as our programming language, along with libraries and tools like:
   - Jupyter notebooks and Jupyter Lab
   - Numpy
   - Scipy
   - Pandas
   - Scikit-learn
   - Matplotlib (or similar plotting libraries like Bokeh)
   - PySpark

   Many of these packages can be installed locally on your laptops by using a distribution like Anaconda - please install the **Python 3.12** compatible Anaconda distribution on your own laptop and ensure that the software above is also installed.
2. You may also use a **browser-based Jupyter notebook interface** called Colaboratory (often just called **Colab**) to develop and run code in lieu of a locally installed software distribution like Anaconda. Colab notebooks are basically Jupyter notebooks that run in the cloud: you can also link them to your Google Drive (use the Rutgers **Scarlet Apps**) and on GCP.
3. You need to be comfortable using the Unix/Linux command line for running code. For Windows users, please install the Windows Subsystem For Linux and learn how to interact with a Linux shell.

Logistics

We will be using the Rutgers **Canvas** site for the course, as well as Github or Google Drive to share class material and assessments. The Discussions space on the Canvas site will be a

good place to share ideas and questions.

There will be a **term project** for the course to be done in groups of 3-4 students: each group should have at least one graduate student and one undergraduate student. The term project will have a few check-points that will include presenting the proposal (by way of a recorded 5-minute flash talk) providing an intermediate update (as a recorded video presentation), and a final in-class presentation (about 10-15 minutes per group). The entire term project will have a timeline of about 9 weeks.

From the second session, I will follow a **semi-flipped** approach which involves **active learning**. Students must be prepared for the session by watching assigned videos, assigned reading material from the textbooks or relevant websites, and experimenting with relevant code discussed in the previous class.

During each active session, a traditional lecture will be complemented with collaborative questioning-answering and short, formative assessments to check your understanding of the concepts. There will also be *timed* quizzes (graded) and coding exercises (also graded) that build on the material: some of these normative assessments will be done in the classroom during the session.

To help students engage more deeply with the content, I will assign additional homework problem sets periodically through the semester. These sets will contain a mix of theory questions and code.

Grading

> **!** Important
>
> Graduate students or undergraduate students in the accelerated Master's program *will have extra assessments*: for instance, they may have to solve an additional problem in a homework problem set, or implement something extra in a coding exercise etc.

The overall grade will be apportioned based on the following assessments (there is no final exam for the course):

- **25%** Four or five homework problem sets (distributed at roughly 2 week intervals)
- **20%** Weekly Quizzes
- **15%** Weekly code exercises
- **20%** **In-class** midterm exam during the last 90 minutes of the **October 16th** session
- **20%** Term Project

Except in the most extenuating circumstances, there will be **no makeup** opportunities for the midterm exam, in-class assessments or the term project check-points. Late submission of homework problem sets will also carry a penalty except in the case of a verifiable, documented emergency (medical or personal).

Letter grade rubric

I use the following grade curve:

- an A grade for cumulative points above 82.5%
- a B+ grade in the range 75 – 82.5%
- a B grade in the range 65 – 75%
- a C+ grade in the range 60 – 65%
- a C grade in the range 50 – 60%
- a D or F grade below 50%

> **!** Important
>
> - Graduate students should be aware that anything below a B grade in two or more courses may affect your graduation and funding status.
> - There is no mechanism for grade improvement via extra work or additional assessments.

Tentative Schedule of Topics

Please note that this schedule is **tentative** and subject to change. Chapters from the two textbooks are specified below as being either from the **MMDS** or **FDS** textbooks. When a chapter is assigned for reading, I will let you know in advance the specific sections that are to be read. Additional notes, slides, readings and videos will be provided or announced as needed.

> **i** Note
>
> The textbooks (especially **FDS**) assume some background in basic calculus, probability, data structures and algorithms. If you feel challenged by specific derivations/concepts, do ask questions in class and on the Canvas discussion forum, and re-learn the material as necessary. We do not have a TA for the class but I will do my best to ensure that you get adequate support to understand difficult material.

Table 2: Weekly Schedule

Dates	Topics	Reading
9/4	• Preliminaries • Python & Scientific Computing • Probability; Hashing • Linear Algebra primer	
9/11 & 9/18	• High-dimensional Spaces • Distance Metrics • Locality-sensitive hashing	MMDS Ch. 3; FDS Ch. 2
9/18	• Probabilistic Sketching	MMDS Ch. 4; FDS Ch. 6
9/25 10/2	• Streaming Algorithms	
10/9 & 10/16	• Matrix decompositions • Singular Value Decomposition • **Midterm Exam**	MMDS Ch. 11; FDS Ch. 3
10/23 & 10/30	• Collaborative Filtering • Clustering	MMDS Ch. 7; FDS Ch. 7
11/6 11/13	• Classical Machine Learning • Support Vector Machines	MMDS Ch. 12; FDS Ch. 5
11/20	• Gradient Descent	
11/27	**No Class Session: Thanksgiving**	
12/4	Deep Learning Basics	MMDS Ch. 13
12/11	Project Presentations	

Academic Integrity

Some of the in-class assessments will be collaborative, while others will need to be completed individually. There are lots of online sites and tools based on Large Language Models (LLMs) that can help you to solve your homework problems, but be warned that there are many gaps in reasoning and incorrect solutions in the wild (including code that you may find on websites or after consulting your favorite LLM assistant). If challenged, you will have to be prepared to clearly explain your code and/or written submissions to my satisfaction.

I strongly encourage you to learn from other books, internet resources like StackOverflow, and even by prompting LLMs like GPT-X/ CoPilot/Bard. But, you must **cite your sources** at all times! Copying from someone or somewhere without citation, or allowing your work to be copied by others is cheating, as is *blind transcription* from sources including books,

LLM transcripts and the internet at large. *You* are ultimately responsible for what you turn in: if it is determined that your work (even with citations) is merely derivative and you haven't completely understood what you have submitted, it will count as an academic integrity violation and may carry pretty serious penalties.

I will follow the Rutgers Academic Integrity Policy and Student Code of Conduct to deal with suspected violations. Please read and understand the policy carefully.