UC Berkeley School of Information | MIDS Program

DATASCI w261
Machine Learning at Scale

Course Architect: Dr. James G. Shanahan, 1/1/2015
Syllabus Last Revised on 8/10/2018.

Course Overview

This 3 Unit course covers the underlying principles required to develop scalable machine learning analyses for structured and unstructured data at the petabyte scale. Content is delivered via asynchronous video lectures, readings from academic textbooks, synchronous session discussions, live code demonstrations, and independent homework assignments.

The course is designed around three goals. By the end of the term students will...

1. … learn to recognize and apply key concepts in parallel computation and map-reduce design.
2. ... design stateless parallelizable implementations of core machine learning algorithms from scratch.
3. ... gain hands-on experience using Apache Hadoop and Apache Spark to analyze large datasets.

Prerequisites: Introductory machine learning course or equivalent. Intermediate programming capabilities in an object-oriented language (such as Python).

Content Description

Until recently, “big data” was very much the purview of database management and summary statistics systems such as Hadoop (HDFS and MapReduce) and was largely underleveraged by machine learning. This course builds on and goes beyond this collect-and-analyze phase of big data by focusing on how machine learning algorithms can be rewritten and in some cases extended to scale to work on petabytes of data, both structured and unstructured, to generate sophisticated models that can be used for real-time predictions. Predictive modeling at this scale can lead to huge boosts in performance (typically in the order of 10–20%) over small-scale models running on stand-alone computers that require one to significantly down-sample and, necessarily, to simplify big data. Concretely, this course focuses on how the map-reduce design paradigm from parallel computing can be extended and more faithfully leveraged to tackle the somewhat “embarrassingly parallel” task of machine learning (a lot of machine learning algorithms fit this mold).

The Apache Spark project and its many related subprojects exemplify the continued relevance of map-reduce style algorithm design. Apache Spark is an open-source cluster-computing framework. It has emerged as the next-generation big data processing engine, overtaking Hadoop MapReduce, which helped ignite the big data revolution. Spark maintains MapReduce’s linear scalability and fault tolerance but extends it in a few important ways: it is much faster (100 times faster for certain applications); much easier to program in, due to its rich APIs in Python, Java, and Scala (and R) and its core data abstraction, the distributed data frame; and it goes far beyond batch applications to support a variety of compute-intensive tasks, including interactive queries, streaming, machine learning, and graph processing.

This course will provide an accessible introduction to MapReduce frameworks and to Spark and its potential to revolutionize academic and commercial data science practices through scale. Conceptually, the course includes two simultaneous components. The first covers fundamental concepts of MapReduce parallel computing via Hadoop and Spark. The second focuses on hands-on algorithmic design and development in parallel computing environments such as Spark; developing algorithms from scratch, such as decision-tree learning; graph-processing algorithms such as PageRank and shortest path; gradient descent algorithms such as support vectors machines; and matrix factorization. Industrial applications and deployments of MapReduce parallel compute frameworks from various fields, including advertising, finance, healthcare, and search engines, help tie these components together. Examples and exercises will be made available in Python notebooks (Hadoop streaming and PySpark).

Weekly Topics

Week	Topic	Deadlines
1	Intro to Machine Learning at Scale	
2	Parallel Computation Frameworks	HW 1 due
3	Map-Reduce Algorithm Design (part 1)	
4	Map-Reduce Algorithm Design (part 2)	
5	Intro to Spark/Map-Reduce with RDDs	HW 2 due
6	Spark Optimizations for Big Data	
7	Distributed Supervised ML (part 1)	HW 3 due
8	Distributed Supervised ML (part 2)	
9	Graph Algorithms at Scale (part 1)	HW 4 due
10	Graph Algorithms at Scale (part 2)	
11	ALS and Spark MLLib	HW 5 due
12	Decision Trees	
13	Spark DataFrames and Online Learning	
14	Special Topics in Distributed ML	Final Projects Due

Assignments and Materials

This is an upper level graduate course. As such, we expect students to exhibit a high level of conscientiousness and initiative in their approach to preparing for class and completing assigned work. Each week you will have assigned video content as well as readings -- both should be completed before your live session*. In live session we will typically review a short set of conceptual slides and/or break in to groups to do a code demonstration. In some cases your live session instructor may ask you to review the first section of a demo notebook before you arrive in class. These code “demos” are designed to set you up for success on the homeworks. They offer a low-risk opportunity to build supporting skills-- the more actively you engage with them during class the less time you will end up spending on the homeworks.

***Note on the 2018 redevelopment:** During the Spring, Summer and Fall terms of 2018 this course is undergoing a rapid iteration process to update the ways we deliver content and transition to an understanding by design pedagogical framework. We thank you for your patience as we work to improve the student experience.

> <u>What this means for you.</u> *As of this (fall 2018) term, the async videos have not yet been revised. We believe the existing videos continue to be a rich source of information and inspiration but in some cases they may not be perfectly aligned to the live session and homework. We expect that you* will *watch them, but when preparing for class you may wish to emphasize the readings over the videos.*

Grading Policy

% of Final Grade	Component
70%	Homework Assignments (5 assignments, 14% each)
20%	Final Project
10%	Live session attendance and participation

Homework

Each independent homework assignment consists of a python notebook with coding and short response conceptual questions. We believe these assignments to be active learning experiences and hope you will approach them as an opportunity to develop your own understanding and not just a source of a grade. Feedback from past students has consistently indicated that the challenge of these assignments is what makes the course valuable to them post-MIDS. For now, here’s what you need to know:

- **Accessing and Submitting:** Your homework will be distributed and submitted via GitHub. (see the “Logistics” section below for details). DO NOT SUBMIT HOMEWORK ON ISVC.
- **Time Commitment:** We expect a well prepared student to spend 15-25 hours on each homework. Depending on your background this time will vary.
- **Late-Day Policy:** As much as we are committed to serving a rigorous course we also know you are hard working professionals, parents, and partners. To give you some flexibility we offer **7 late days** that you can use to <u>extend any homework deadline by up to 2 days</u>.
- **Partial Credit:** We grade each multi-part question holistically; always attempt as much as you can because we’d love to give you partial credit for partial understanding.

Final Project

The final project is a group assignment that offers an opportunity to demonstrate your mastery of the course goals. The scope of the project is similar to that of the homeworks (we will provide a dataset and assign an algorithm) but unlike the homework assignments we will not provide any code base. We will assign groups and release a rubric in week 9 and your team will have one month to organize the workload, perform relevant EDA, implement the algorithm, and deliver a python notebook based analysis report including citations results and discussion of parallelization concepts that impacted design choices you made.

Readings

This course will use a combination of textbook chapters and some online readings. We have provided PDF copies of chapters for open sourced materials but for the starred items below students will need to borrow or buy their own copies.

Recommended Textbooks:

- *Karau, Holden, Konwinski, Andy, Wendell, Patrick, & Zaharia, Matei. (2015). *Learning Spark: Lightning-fast big data analysis*. Sebastopol, CA: O'Reilly Publishers.
- Lin, Jimmy, & Dyer, Chris. (2010). *Data-intensive text processing with MapReduce*. San Rafael, CA: Morgan & Claypool Publishers. (Free online)
- Karau, Warren. (2017). *High Performance Spark: Best Practices for Scaling and Optimizing Apache Spark.* Sebastopol, CA: O'Reilly Publishers.
- Hastie, Trevor, Tibshirani, Robert, & Friedman, Jerome. (2009). *The elements of statistical learning: Data mining, inference, and prediction* (2nd ed.). Stanford, CA: Springer Science+Business Media. (Free online)
- Hastie, Trevor, Tibshirani, Robert, Witten, Daniela, & James, Gareth. (2014). *An Introduction to Statistical Learning: with Applications in R*. Stanford, CA: Springer Publishing Company. (Free online)
- *Ryza, Sandy, Laserson, Uri, Owen, Sean, & Wills, Josh. (2015). *Advanced analytics with Spark: Patterns for learning from data at scale*. Sebastopol, CA: O'Reilly Publishers.
- Leskovec Jure, Rajaraman Anand, Ullman Jeff, (2014). *Mining of Massive Datasets*, Cambridge University Press. Book available online at http://www.mmds.org/

Logistics and Communication

GitHub is the source of ground truth for assignments, deadlines and policies in this course. In the first week of class you will create your own personal w261-homework repository where you will submit assignments. We will rely on Slack to communicate about any changes, discuss content, and provide clarification. However if you need to get in touch with an instructor for a non-content related reason please use email and cc the TA for your section.

GitHub Repo: https://github.com/UCB-w261/main

The README.md file in this repo contains your weekly reading assignments, class schedule, homework deadlines and links to all materials you'll need for live session. This is also where we'll release homework assignments. You will set up this repo as an upstream remote on your personal repository to pull down the templates each week. (*ask a TA in week 1 if you need help with this*)

Slack Channels

Slack serves a few purposes for this course. In the first week of classes you will receive invites to two private channels, one for general discussion (**#w261-fall2018**) and one for troubleshooting infrastructure issues related to the Docker container or GCP cluster (**#w261-infrastructure**).

To keep these channels helpful for everyone please follow a few norms:

- Use threaded replies to help keep different lines of conversation easy to find.
- Respect that everyone comes from a different background -- share your silly questions freely and answer others' queries with good cheer.
- When asking a question, describe what you've already tried to resolve your own question and reference any course materials you may be looking at -- this helps us make better suggestions faster.
- Commiserating can be a form of support (especially in a very time consuming class) but watch out for the line where commiserating turns into complaining -- don't bring or put others down and use other channels (eg. surveys or email) to share constructive criticism intended for instructors ears.

Instructor Slack handles: **kylehamilton**, **mike.tamir**, **mike-bowles**, **mmillervedam, davehuber, jameswine**

Office Hours

Office hours will be held every week. Refer to the course repo for dates/times that each instructor will be available.

Infrastructure

Although learning how to set up a distributed environment for parallel computation is *not* a learning objective for this course, many of the concepts that are central to understanding parallel computation *do* require students to learn a little bit about infrastructure. We hope to keep infrastructure frustrations to a minimum by providing two consistent environments for students so that we can help you navigate reproducible errors related to your environment configuration:

- **Docker Container**: You will download and set up the course docker container as part of the 1st homework assignment. If you haven't use Docker before you will need to install it. You will need 4 cores, 8 GB of RAM, 30 GB of disk space and Linux, Mac, or Windows 10 Pro/Enterprise/Education to run the course container and complete the homeworks. This container includes Hadoop and Spark distributions. You will want to mount the local directory where you've cloned your personal repository so that you can work inside the container and push homeworks from there. More information can be found in the course environment repo (see HW1).
- **GCP DataProc**: We also provide access to a course cluster. For the larger homework assignments you will submit your spark jobs using a custom script that we provide. More information in live session 5.

In the past, students with the skillset to do so have set up their own environment (eg. using AWS or Databricks). While you are welcome to do so, you should be aware that Hadoop and Spark may have different out-of-the box behavior depending on your system and we cannot support you directly unless you use the provided course materials.