Syllabus for STA 663

Instructor:: Cliburn Chan cliburn.chan@duke..edu
Instructor: Janice McCarthy janice.mccarthy@duke.edu
TA: Matt Johnson mcj15@stat.duke.edu

Overview

The goal of STA 663 is to learn statistical programming - how to write code to solve statistical problems. In general, statistical problems have to do with the estimation of some characteristic derived from data - this can be a point estimate, an interval, or an entire function. Almost always, solving such statistical problems involves writing code to collect, organize, explore, analyze and present the data. For obvious reasons, we would like to write good code that is readable, correct and efficient, preferably without reinventing the wheel.

This course will cover general ideas relevant for high-performance code (data structures, algorithms, code optimization including parallelization) as well as specific numerical methods important for data analysis (computer numbers, matrix decompositions, linear and nonlinear regression, numerial optimization, function estimation, Monte Carlo methods). We will mostly assume that you are comfortable with basic programming concepts (functions, classes, loops), have good habits (literate programming, testing, version control) and a decent mathematical and statistical background (linear algebra, calculus, probability).

To solve statistical problems, you will typically need to (1) have the basic skills to collect, organize, explore and present the data, (2) apply specific numerical methods to analyze the data and (3) optimize the code to make it run acceptably fast (increasingly important in this era of “big data”). STA 663 is organized in 3 parts to reflect these stages of statistical programming - basics (20%), numerical methods (60%) and high performance computing (20%).

Learning objectives

The course will focus on the development of various algorithms for *optimization* and *simulation*, the workhorses of much of computational statistics. The emphasis is on *comptutation* for statistics - how to prototype, optimize and develop high performance computing (HPC) algorithms in Python and C/C++. A variety of algorithms and data sets of gradually increasing complexity (1 dimension → many dimensions, serial → parallel → massively parallel, small data → big data) will allow students to develop and practise the following skills:

- Practices for reproducible analysis
- Fundamentals of data management and munging
- Use Python as a language for statistical computing
- Use mathematical and statistical libraries effectively
- Profile and optimize serial code
- Effective use of different parallel programming paradigms

Pre-requisites

Review the following if you are not familiar with them

- Unix commands
- Using `git` for version control
- Writing Markdown
- Writing $\LaTeX$
- Using `make` to build programs

The course will cover the basics of Python at an extremely rapid pace. Unless you are an experienced programmer, you should probably review basic Python programming skills from the Think Python book. This is also useful as a reference when doing assignments.

Another very useful as a reference is the official Python tutorial

Grading

- Quizzes (25%)
- Programming project 1 (25%)
- Programming project 2 (25%)
- Final exam (25%)

Computing Platform

Each student will be provided with access to a virtual machine image running Ubuntu - URLs for inidividual students will be provided on the first day. For GPU computing and map-reduce examples, we will be using the Amazon Web Services (AWS) cloud platform. Again, details for how to acccess will be provided when appropriate.

All code developed for the course should be in a personal Github repository called sta–663-

firstname-lastname. Make the instructors and TA collaborators so that we have full access to your code. We trust that you can figure out how to do this on your own.

Lecture 1

- The IPython notebook
    - Markdown cells
    - Code cells
    - The display system
    - IPython magic
    - Interfacing with other languages
- Programming in Python
    - Basic types
    - Basic collections (tuples, lists, sets, dicts, strings)
    - Control flow
    - Functions
    - Classes
    - Modules
    - The standard library
    - PyPI and `pip`
    - Importing other modules

Lecture 2

- Functional programming
    - Functions are first class objects
    - Pure functions
    - Iterators
    - Generators
    - Anonymous functions with lambda
    - Recursion
    - Decorators
    - The `operator` module
    - The `itertools` module
    - The `functools` module
    - The `toolz` module
    - Constructing a lazy data pipeline
- Working with text
    - string methods

    - The `string` module
    - The `re` module

Computer lab 1

- Exercise 1: Generating a report with `make`, $\LaTeX$ and python
- Exercise 2: Functions to calculate mean, variance and Pearson correlation coefficient
- Exercise 3: Project Euler puzzle 1
- Exercise 4: Constructing a functional pipeline using generators

Lecture 3

- Obtaining data
    - CSV with `csv`
    - JSON with `json`
    - Web scraping with `scrapy`
    - HDF5 with `pyhdf`
    - Relational databases and SQL with `sqlite3`
    - The `datasets` module
- Scrubbing data
    - Removing comments
    - Filtering rows
    - Filtering columns
    - Fixing inconsistencies
    - Handling missing data
    - Removing unwanted information
    - Derived information
    - Sanity check and visualization

Lecture 4

- Using `numpy`
    - Data types
    - Creating arrays
    - Indexing
    - Broadcasting
        - Outer product

    - Ufuncs
    - Generalized Ufuncs
    - Linear algebra in numpy
        - Calculating covariance matrix
        - Solving least squares linear regression
    - I/O in numpy
- Using `pandas`
    - Reading and writing data
    - Split-apply-combine
    - Merging and joining
    - Working with time series
- Using `blaze`

Computer lab 2

- Exercise 1: Make a 12 by 12 times table chart without looping
- Exercise 2: Working with CSV, JSON, HDF5 and RDBMS data
- Exercise 3: Working with some data set in `pandas`
- Exercise 4: Plotting the scatter matrix for the Iris data set in `matplotlib`, `seaborn` and `bokeh`

Lecture 5

- From math to computing
    - Computer representation of numbers
    - Overflow, underflow, catastrophic cancellation
    - Stability
    - Conditioning
    - Direct translation of symbols to code is dangerous
- The purpose of computing is insight not numbers
- Examples of computation in statistics
    - Estimating parameters (point and interval estimates)
    - Estimating functions
    - Feature extraction, class discovery and dimension reduction
    - Classification and regression
    - Simulations and computational inference
- Algorithmic efficiency and big $\mathcal{O}$ notation
    - Examples from classic data structures and algorithms

Lecture 6

- Numerical linear algebra
  - Simultaneous linear equations
  - Column space, row space and rank
  - Rank, basis, span
  - Norms and distance
  - Trace and determinant
  - Eigenvalues and eigenvectors
  - Inner product
  - Outer product
  - Einstein summation notation
- Matrices as linear transforms
  - Types of matrices
    - Square and non-square
    - Singular
    - Positive definite
    - Idempotent and projections
    - Orthogonal and orthonormal
    - Symmetric
    - Transition
  - Matrix geometry illustrated

Computer lab 3

- Exercise 1: Project Euler puzzle 2
- Exercise 2: Project Euler puzzle 3
- Exercise 3: Project Euler puzzle 4
- Exercise 4: Project Euler puzzle 14

Lecture 7

- Matrix decompositions
  - LU (Gaussian elimination)
  - QR
  - Spectral
  - SVD
  - Cholesky

- Using `scipy.linalg`
- BLAS and LAPACK

Lecture 8

- Projections, ordination, change of coordinates
    - PCA in detail
    - PCA with eigendecomposition
    - PCA with SVD
    - Related methods
        - ICA
        - LSA
        - Factor analysis

Computer lab 4

- Exercise 1: Solving a least squares problem using Cholesky decomposition
- Exercise 2: Implement latent semantic indexing
- Exercise 3: Implement k-means clustering
- Exercise 4: Use latent semantic indexing to reduce a set of documents to 2D then use k-means to cluster them, and finally plot the result

Lecture 9

- Regression and maximum likelihood as optimization problems
- Local and global extrema
- Univariate root finding and optimization
    - Golden section search
    - Bisection
    - Newton-Raphson and secant methods
- Packages for optimization
    - Using `scipy.optimization`
    - Using `statsmodels`
    - Using `scikits-learn`
- General approach to optimization
    - Know the problem
    - Multiple random starts

    - Combining algorithms
    - Graphing progress

Lecture 10

- Multivariate optimization
    - Non-gradient based (Nelder-Mead)
    - First order (gradient descent and stochastic gradient descent)
    - Second order (Newton's mehtod)
    - Variations on Newton's methods (IRLS, conjugate gradiet, LM)

Computer lab 5

- Exercise 1: Given a sequence of function values, write a program to perform kernel density estimation using several kernels
- Exercise 2: Use line search optimization to solve a 1D logistic regression problem
- Exercise 3: Implement the secant method for finding roots in 1D
- Exercise 4: Implement Newton's method and find an approximate solution to several equations (including ones that diverge)

Lecture 11

- Constrained optimization
    - Lagrange multipliers
    - Barriers and penalties
    - Transfomations
    - Convex optimization
    - Packages: `cvxopt`, `cvxpy` and `pyopt`

Lecture 12

- The EM algorithm (1)
    - Convex and concave functions
    - Jensen's inequality
    - Missing data setup

    - Toy example - coin flipping with 2 biased coins

Computer lab 6

- Exercise 1: Write the SGD function to solve a multivariate logistic regression problem using maximum likelihood
- Exercise 2: Write the EM algorithm to solve another toy problem
- Exercise 3: Using scipy.optimize
- Exercise 4: Using scikits-learn

Lecture 13

- The EM algorithm (2)
    - Gaussian mixture model
    - EM for Bayesians - MAP of posterior distribution
    - Other applications of EM
    - EM variants

Lecture 14

- Monte Carlo methods
    - Random number generators
    - Generating random variates from a distribution
    - Quadrature, Monte Carlo estimation and Monte Carlo swindles
    - Estimate confidence intervals (bootstrap)
    - Compare competing statistics (statistical simulation - e.g. power)
    - Compare models (cross-validation)
    - Hypothesis testing (permutation-resampling)

Computer lab 7

- Exercise 1: Modify the EM algorithm for GMMs to find the MAP estimate of the posterior distribution
- Exercise 2: Use k-fold cross-validation to evaluate which is the best model for a given data set
- Exercise 3: Estimate the distribution of the slope in a linear regression model by bootstrapping on the residuals

- Exercise 4: Find the type–1 error for $\alpha = 0.05$ by using permutation resampling to correct for multiple testing

Lecture 15

- Conducting a simulation experiment (case study)
- Experimental design
    - Variables to study
    - Levels of variables (factorial, Latin hypercube)
    - Code documentation
    - Recording results
    - Reporting
    - Reproducible analysis with `make` and $\LaTeX$

Lecture 16

- MCMC (1)
    - Toy problem - rats on drugs
    - Monte Carlo estimation
    - Importance sampling
    - Metropolis-Hasting
    - Gibbs sampling
    - Hamiltonian sampling
    - Assessing convergence
    - Using `pystan`
    - Using `pymc2`
    - Using `emcee`

Computer lab 8

- Exercise 1: Writing a Gibbs sampler for change point detection
- Exercise 2: Using `pystan`
- Exercise 3: Using `pymc2`
- Exercise 4: Using `emcee`

Lecture 17

- MCMC (2)
    - Gaussian mixture model revisited
    - Gibbs sampling
    - Infinite mixture model with the Dirichlet process

Lecture 18

- Profiling
    - Premature optimization is the root of all evil
    - Using `%time` and `%timeit`
    - Profiling with `%prun`
    - Line profiler
    - Memory profiler
- Code optimization
    - Use appropriate data structure
    - Use appropriate algorithm
    - Use known Python idioms
    - Use optimized modules
    - Caching and memorization
    - Vectorize and broadcast
    - Views
    - Stride tricks

Computer lab 9

- Exercise 1: The label-switching problem
- Exercise 2: Classifying points with the GMM:
- Exercise 3: Profiling source code
- Exercise 4: Optimizing source code

Lecture 19

- JIT compilation with `numba`
- Optimization with `cython`

- Wrapping C code
- Wrapping C++ code
- Wrapping Fortran code

Lecture 20

- Why modern CPUs are starving and what can be done about it
- Parallel programming patterns
- Amdahl's and GustClassifying points with the Gustafson's laws
- Parallel programming examples
    - JIT compilation with `numba`
    - Toy example - fractals
    - Using `joblib`
    - Using `multiprocessing`
    - Using `IPython.Parallel`
    - Using `MPI4py`

Computer lab 10

- Exercise 1: Optimizing EM code with numba
- Exercise 2: Optimizing EM code with Cython
- Exercise 3: Parallel processing of embarrassingly parallel code
- Exercise 4: Parallel processing of code requiring intra-process communication

Lecture 21

- GPU computing
    - Introduction to CUDA
    - Vanilla matrix multiplication
    - Matrix multiplication with shared memory
    - JIT compilation with `numba`
    - Example: Large-scale GMMs with CUDA

Lecture 22

- Map-reduce and Spark (AWS)
    - Problem - k-mer counting for DNA sequences
    - Small scale map-reduce using Python
    - Using `hadoop` with `mrjob`
    - Using `spark` with `pyspark`
    - Using `MLib` for large-scale machine learning

Computer lab 11

- Exercise 1: Coding fractals in CUDA
- Exercise 2: Large-scale GMMs with CUDA
- Exercise 3: Word count in map-reduce
- Exercise 4: K-mer count with map reduce with E Coli and human genome

Data sets
- ecoli genome
- human genome

Supplementary Mateiral

SM 1

- Using the command line
    - The Unix philosophy and `bash`
    - Remote computing with `ssh`
    - Version control with `git`
    - Documents with $\LaTeX$
    - Automation with `make`

SM 2

- Graphics in Python
    - Using `matplotlib`
    - Using `seaborn`
    - Using `bokeh`