Comments on CS401/501: Introduction to Data Science

Geoffrey Exoo

February 9, 2021

You will find that most students are ill-prepared in either Math or CS, or both. So you will have to spend much of the semester teaching Python and reviewing parts of Calculus and Linear Algebra. For the Math, I try to show examples that depict what gradients and various vector products mean visually, rather than expect them to really understand the Math. Similarly for topics in Linear Algebra, particularly eigenvectors and eigenvalues. I've tried using R instead of Python, but teaching them to do any real programming in R (as opposed to just using modules) takes too much time.

I. Topics outside the main flow of the course that I usually spend time on.

- A) Linear Algebra: matrix multiplcation, dot products, cross products, eigenvectors, eigenvalues, etc.
- B) Calculus: vectors, gradients.
- C) Probability and Counting.
- D) Python.
  - 1) Comprehensions, Iterators, Generators.
  - 2) Choosing appropriate data structures from among lists, tuples, sets, frozensets, and dictionaries.
  - 3) Performance related modules: time, timeit, dis
  - 4) Data Science Modules: numpy, pandas and matplotlib. Llearning these three modules could be the whole course, something I try to prevent.
  - 5) Regular expressions, in Python and in general.
  - 6) Writing Python modules in C (for speed). This worked better back when they knew C.
  - 7) Other useful modules: itertools, collections.

II. Data Cleaning and Formatting (actually the first topic)

- A) Topics.
  - 1) Extracting data from files: command line tools `grep`, `sed`, `awk`, etc.
  - 2) Repairing bad data: Python modules `re` or `regex`, `numpy`
- B) Projects.
  - 1) Author Identification - Step 1.

    a) Download a few Project Gutenberg novels (text files) by each of a list of authors (17th and 18th century British authors are good choices).
    b) Use command lines tools to parse the text files and generate lists of word frequencies for each author.
    c) Use Python to do the same job, and the automate the process of removing material added by Project Gutenberg (copyright, table of contents, footnotes, etc.) so only the authors words are counted.
  2) Data Cleaning with Numpy - Weather Data
    a) Download historical weather data for a few cities (Terre Haute, Indianapolis, etc.) from Illinois Weather Center.
    b) Use `numpy` and various interpolation methods to fill in missing data.
    c) Repair bad data (e.g., nonzero snowfall in July with temperatures in the 80's).

III. Data Presentation.

- A) Topics.
  1) Python Modules: `matplotlib`
- B) Projects.
  1) Author Identification - Step 2.
    a) Pick two words (e.g., "could and would"), compute frequencies for a list of books, use frequencies as coordinates for 2D plotting.
    b) Same thing for three words (e.g., "could, should and would") and do 3D plotting.
    c) You could spend a semester on `matplotlib`.
  2) COVID-19 Data.
    a) The New York Times data set is very easy to use.
    b) Plot things that require some calculation, e.g., running totals of cases per week.

IV. Classifiers.

- A) Classification Algorithms.
  1) k-Nearest Neighbor (good intro, everyone understands it)
  2) Decision Trees and Random Forests (also popular, very little Math)
  3) Naive Bayes (requires Probability review)
  4) Logistic regression (logs and exponentials). This can be viewed as an introduction to both SVMs and ANNs, the next two topics.
  5) Support Vector Machines (vectors, dot products, some Calculus)
  6) Artifical Neural Networks (I omit them here, do them separately)
  7) Save the fancy stuff for 601.
- B) Projects.
  1) Author Identification - Step 3.

    a) Pick N common words ($2 \leq N \leq 100$). Then for each author compute frequencies of these words. View results as points in N-space. Then do the same thing for a few new books, author unknown (to the student), and apply k-Nearest Neighbor to determine which author wrote the book. Works great with "could, should, would" and Jane Austen vs. Charles Dickens.
    b) Same idea, but using Naive Bayes. Good exercise in understanding how to deal with floating point round-off errors. With $N = 50$, it produces impressive results (surprised me anyway).
  2) Character recognition.
    a) Use the MNIST digit data set (the standard).
    b) Apply k-nearest neighbor and see how well it does. It will not work as well as later methods, but will easily beat random guessing, and sets a baseline that you can improve with other methods.

V. Dimensionality Reduction. This is certainly one of the most, if not the most important topic, but it requires too much Math. So I give it a couple of days.

- A) Principal Component Analysis.
  1) Eigenvalues and eigenvectors: Covariance matrix demo.
  2) Hopefully, I get them to understand the standard 2D data plot (the fuzzy ellipse where one dimension is clearly more important that the other).

VI. Deep Learning and Artificial Neural Networks (ANNs)

- A) Topics.
  1) The Calculus behind Gradient Descent (maybe just in pictures).
  2) The Linear Algebra behind ANNs (matrix multiplication, outer products, inner products).
  3) Build a fully connected two layer ANN "by hand" in Python that works for a small "fake" data set.
  4) Build a full featured fully interconnected multilayer ANN "by hand" in Python. Apply it to MNIST data.
  5) Introduce convolutional networks, resorting to using packages like Pytorch, XGBoost, etc. Hard to stay current in this stuff. Pytorch is still popular and easy to use.
- B) Projects.
  1) Character recognition.
    a) Again use the MNIST digit data set.
    b) Use a two-level fully connected network. Will get about 80 percent correct. Experiment with learning rates.
    c) Use a three-level fully connected network. Will get about 90 percent correct. Experiment with learning rates and activation functions.
    d) Use Pytorch and convolutional layers. Can achieve 95 to 99 percent accuracy. Good spot to introduce GPU programming.

VII. Other Topics That I've Tried. Probably more appropriate for 601.

A) Sentiment Analysis (Product Reviews). Advantages: there is a lot of good data available. Disadvantages: the methods that work are rather complex.

B) Associative Learning (Market Basket Analysis). Disadvantages: good data is hard to find. Advantages: good application and comparison of breadth first search and depth first search.

C) Natural Language Processing. Same comments as for sentiment analysis.