Course Administration

Prof. Phil Schniter

THE OHIO STATE UNIVERSITY

ECE 5307: Introduction to Machine Learning, Sp23

Instructional team

- Prof. Phil Schniter (schniter.1@osu.edu)
  - Office hours in Dreese 616 and on zoom: Th 12-2pm
  - Zoom link for Prof. Schniter's office hours
- Ms. Kaiying Xie (xie.916@buckeyemail.osu.edu)
  - Will grade homework and labs
  - Office hours in Dreese 331 (no zoom): We 6:30-8pm
Th 4-8pm
  - Office hours zoom-only: Fr 11am-1pm
  - Zoom link for Kaiying's office hours

Course learning objectives

- Understand what machine learning is
  - Regression, classification, clustering, dimensionality reduction, etc.
- Understand the *mathematics* behind standard models, algorithms, and methods
  - Including concepts from linear algebra, probability, and statistics.
- Gain experience with standard coding tools:
  - Python, Numpy, scikit-learn, Jupyter, matplotlib
  - PyTorch
  - GitHub

Prerequisites

- Undergrad in ECE or graduate standing (in any major)
- Calculus and Linear Algebra: Math 2568 / Math 4568, or equivalent
  - Vectors, matrices, eigenvectors, partial derivatives, gradients
  - Homework 0 is a self-guided review! We will do additional review as needed
- Probability and Statistics: Stats 3470 / Math 4530, or equivalent
  - Basic concepts such as mean, variance, correlation, probability densities, conditional distributions, Gaussian distribution
  - We will review as needed
- Programming: CSE 1222 or ENGR 1281, or equivalent
  - Basic scientific programming such as C, C++, Matlab, or Python
  - We will teach you Python; no previous background is assumed

Course delivery mode = "in person"

- I will lecture in Journalism 251 from 4:10-5:05pm every MWF
  - Attendance is expected
  - Lecture pdfs (such as this one) will be posted in advance
  - Bring a copy of the pdf to the lecture so you can take notes and fill in any blanks!
  - Lecture recordings will be posted on this YouTube channel
  - In the rare case that lectures need to be given online, we'll use this zoom link
- The TAs and I will hold regular office hours (see page 2)
- Carmen Discussions: post your questions and get quick answers
  - Please post, rather than email, your questions (unless of a personal nature)
  - Students are encouraged to post answers!
  - The TAs and I will answer questions as needed, but please be patient

Grading

- Components:
  - Weekly analytical homeworks: 20% ...due Fridays @ 4pm
  - Weekly coding labs: 20% ...due Fridays @ midnight
  - Weekly Carmen quizzes: 20% ...due Fridays @ 4pm
  - Two midterms: 20% ...tentatively 2/7/23 and 3/31/23
  - Final Project: 20% ...tentatively due 4/25/23
- Final letter grade:
  - Will be curved based on your cumulative score
  - Undergrads will be curved differently than grad students
  - Histograms will show where you stand relative to peers

Final project

- Goal: apply machine-learning concepts taught *in this class*
  - You'll be given a dataset and a design objective
  - You'll apply knowledge from this class to design/implement solutions
  - An in-class kaggle competition will motivate/calibrate your performance
- Projects are team-based, with 3-person teams
  - You can choose your partners
  - Or you can let us choose them for you
- Deliverables: Python code and written report
  - Detailed instructions will be provided later in the term

Textbooks (all optional)

1. G. James, D. Witten, T. Hastie, and R. Tibshirani, *An Introduction to Statistical Learning*, 2013.
   - Free at `https://hastie.su.domains/ISLR2/ISLRv2_website.pdf`
   - Excellent for theory, but code examples in R (not Python)
2. C. Albon, *Machine Learning with Python Cookbook*, 2018.
   - Excellent summary of Python commands, but no theory/concepts
3. A. Géron, *Hands-On Machine Learning with Scikit-Learn, Keras & TensorFlow*, 2019.
   - Good blend between concepts and Python coding examples
   - We'll be using scikit-learn extensively, but not Keras or TensorFlow
4. E. Stevens and L. Antiga, *Deep Learning with PyTorch*, 2020.
   - `https://www.manning.com/books/deep-learning-with-pytorch` but you can usually find a free pdf via google search
   - We'll be using PyTorch for deep learning (in place of Keras and TensorFlow)

CarmenCanvas

- Will be used for ...
  - announcements (make sure your CarmenCanvas notifications are enabled!)
  - discussion boards
  - homework: assignments, pdf submission, solutions, histograms
  - labs: links to GitHub repos, solutions, histograms
  - quizzes
  - exams: solutions, histograms
  - final project: instructions, pdf submission, histograms
  - gradebook: grades, histograms

GitHub

- Will be used for . . .
  - lectures: pdfs
  - demonstration code (via Jupyter notebooks)
  - labs: individual code repositories and submissions
  - final project: group code repositories and submissions
- Primary course repository: `https://github.com/ece5307sp23/ece5307`
  - This is "private"; to get access, follow instructions in CarmenCanvas announcement
  - You can *view* pdfs & Jupyter notebooks with a web browser. But to *run* notebooks, you'll need to clone the repo to your local machine (and keep it updated)
  - To clone & update, you'll need to run GitHub Desktop or git on your local machine. See instructions in the Carmen announcement & primary course repo
- Personal GitHub repos will be created for each lab, and group repos for the final project. You'll need to clone and submit your modified repo.

Python

- Python
  - A general and powerful language
  - Includes many machine-learning libraries, especially for deep learning
  - Free and open source!
- Prerequisites
  - No Python background is assumed; we'll teach it in this course
  - Basic programming experience is required, though (e.g., C or Matlab)
  - Experience with objected-oriented programming (e.g., C++) is helpful
- The primary course repo includes links to ...
  - Tutorials on basic use of Python and Numpy
  - Moving from Matlab to Python (useful if you are experienced with Matlab)
  - Practice materials for Python and Numpy

Python environments

- The labs in this course will use Python and relevant libraries
  - Python 3.x (not 2.x!)
  - Numpy, scikit-learn
  - Jupyter notebooks
  - PyTorch (later in the course)
- You can run Python on your personal machine or in the OSU computer labs
  - I strongly recommend installing Python via the Anaconda distribution
  - Our demos and labs will not require huge computational resources
- If you need more processing power, you can run Jupyter notebooks on GPUs in the cloud. Instructions can be found by googling, e.g.,
  - "How to run Jupyter Notebooks in the cloud"
  - "Six easy ways to run your Jupyter Notebook in the cloud"
  - "The 4 best Jupyter Notebook environments for deep learning"

Jupyter notebook demos

- The lectures will show demonstration code from Jupyter Notebooks, and the notebooks themselves (.ipynb files) can be found in the primary course repo.
- For example, there's `demo00a.ipynb` on how to get started with Numpy:

Getting Started with Numpy Vectors

The `numpy` package has a number of powerful and fast tools for manipulating vectors. In this demo, we will illustrate some of the features of the package that will be used throughout the class. A more complete summary of `python` and `numpy` can be found at:

https://docs.python.org/3/tutorial/

https://docs.scipy.org/doc/numpy/user/quickstart.html

http://cs231n.github.io/python-numpy-tutorial/

For this tutorial, we start by importing the `numpy` package.

```
import numpy as np
```

Creating vectors

We can create vectors in several ways. First, we can manually create a vector by specifying its elements. Note that, unlike MATLAB, there is no difference between row and column vectors. Also, there are no semicolons; you have to call the `print` command to print the object.

```
x = np.array([1,2,4])
print(x)
```

```
[1 2 4]
```

You can also create a vector from a sequence of integers:

```
x1 = np.arange(10)   # numbers from 0 to 9 (note 10 is NOT included)
x2 = np.arange(2,7)  # numbers from 2 to 6 (note 7 is NOT included)
print("x1 = "+str(x1))
print("x2 = "+str(x2))
```

```
x1 = [0 1 2 3 4 5 6 7 8 9]
x2 = [2 3 4 5 6]
```

Jupyter notebook demos (cont.)

- There's also demo00b.ipynb on Numpy's axis and broadcasting features:

Numpy Array Operations: Axes and Broadcasting

There is an excellent introduction to `numpy` multi-dimensional arrays on the scipy website. In this note, we cover two concepts in a little more detail:

- Using the `axis` feature
- Python broadcasting

We will need both of these for performing many of the numerical operations for the ML class.

As usual, we begin by loading the `numpy` package.

```
import numpy as np
```

Axis Parameter

Many operations in the `numpy` package can take an optional `axis` parameter to specify which dimensions the operation is to be applied. This is extremely useful for multi-dimensional data. To illustrate the `axis` parameter, consider a matrix the `(3,2)` array `X` defined as:

```
X = np.arange(6).reshape(3,2)
print(X)
```

```
[[0 1]
 [2 3]
 [4 5]]
```

Questions

Questions?