ML Spring 2017 – Rec. 1

Nishant Mohanchandra

OH: 1:00 – 3:00 PM, MoWe

Agenda

- Quick overview of **numpy** module
- Quick overview of **PS1 – Linear Perceptron**
  - Notes on some common pitfalls
- 30 – 45 minutes of coding time
  - Autumn & I will be on hand for any issues

Homework Submission

- Will be handled via NYU Classes
    - ML on NYU Classes is set up. Check for problem sets under "Assignments"
    - Class resources (lecture / lab slides, notes) will be posted under "Resources"

- Problem Set hand-ins will consist of:
    - All code you worked on (Python)
    - Attached write-up, if any (Templates provided when appropriate)

- Discussion is fine, but final work must be original
    - Clear copying will be looked for and will receive zero credit + a warning

- 7 Problem Sets, each worth ~5% of the grade

Having the correct modules

- Will be useful to have **numpy**, **scipy**, **sklearn** & **matplotlib** for the class
  - Mac installation: **pip3 install [module name]**
  - Windows: Set up your PATH environment variable, and then **pip install [module name]**

- If you are working in Spyder / Anaconda, these should already be installed & ready to go (check by trying import in the shell)

Numpy

- Main object is the homogenous multidimensional array
    - Like a list, but restricted to numeric values
    - Optimized for quick vector and matrix calculations

- Significantly quicker than using raw lists

- For PS1, **np.array**, **np.zeros**, **+** or **-** and **np.dot** should be sufficient

Quick numpy reference

- np.array(sequence) instantiates an array
- np.zeros(dimension[s]) instantiates a zero vector or array
- * for element-wise scalar multiplication
- + and – for vector addition / subtraction
- np.dot(a,b) for the dot product of vectors a and b

More info

- There's support for applying universal functions on a vector element-wise
  - (np.exp, np.sqrt, etc.)
- Indexing, slicing and iteration work the same as with lists
- There's support for splitting, stacking and copying arrays quickly
- Also support for manipulation (diagonals, etc.), plotting, basic linear algebra (cross, etc.)
- Full Quickstart Reference for more info

PS1 – Linear Perceptron

Q1

- Splitting the data is the pre-processing reqd.
  - Slice the 5000 line *spam_train.txt* into a 4000-len training set & 1000-len validation set

- Make sure to separate the labels (1, 0) from the actual e-mail bodies
  - Also, change all the zeros to -1

- Your vocabulary of frequent words (X=20) should only be built from the *training set*
  - *NOTE: the word has to appear in 20 separate emails, not 20 times in the same e-mail*

- Using a dictionary to build the vocabulary can be helpful

Q2

- Don’t worry too much about following “specifications”
  - There is no fixed spec. It just needs to work and make sense
- Just ignore the part about “bias” for now
  - Keith may talk about that later in class
- Cycle through the e-mails in order and update w at every “mistake” (not after a pass of the whole training set)
- Remember that w starts as the zero vector

Q3

- A “mistake” is whenever the sign (+ or -) of the dot product of your current w and the feature vector of the i-th e-mail does not match the given label for that e-mail

- For reference: after you’re done training with N=4000 and X=20…
  - Your training error should be zero (obvious)
  - Your vaidation error should be somewhere around 2%

Q4, Q5, Q6

- Positive weights are correlated with spam and negative weight are correlated with not-spam
  - Remember, each "feature" in your feature vector corresponds to some actual word in your vocabulary
- As a broad trend…
  - Validation error will tend to decrease as N goes up
  - Number of perceptron passes may not have any clear relationship with N
- Excel plots / Python plots / hand-drawn plots etc. are all fine, but must be included in the write-up

Q7, Q8, Q9, Q10

- Q7 involves adding a pass limit to your perceptron implementation
  - When does it come into play?

- For Q8, N=5000 now. This is where the test set in spam_test.txt should come into play

- For Q9: At what X do you think the data is no longer linearly separable?

- For Q10: You should have figured out why validation and testing are important and separate by now

Tricks to try

- Vocabulary generation
  - The initial vocabulary is invariant for a given value of **N**
  - Rather than re-do this each time...
    - Do it once, save the results to a file, and on subsequent runs, just read the file

- The refined vocabulary also is invariant for a given value of **N** and **X**
  - Same trick for vocab generation, with the caveat of an extra variable involved

- Perceptron training
  - Lists are an unoptimized data structure for vector maths. Numpy arrays are faster and provide inbuilts for vector addition and vector dot/ scalar products
  - Runtime constants: refactor code to support command line args

Notes on the test/debug phase

- You always read the e-mails in the same order
  - Results are therefore *consistent* assuming you don't introduce randomness
  - Inconsistent results are a potential red flag

- Some of the "waiting for my code to run" bottlenecks
  - **Generating a vocabulary** (4000 e-mail splits + word-by-word processing)
  - Refining the vocabulary (processing ~50000+ words)
  - **Perceptron training** (4000 vector dot products ***per pass*** + update overhead)
  - Having to manually update runtime constants between tests (N, X)

- Strategies for improvement depend upon sidestepping bottlenecks
  - Can't do much to speed up training, but can avoid vocab generation overhead