CS 335 Homework 2

Last Updated: March 25, 2019

Instructions

Due Friday 2/15 at 11:59pm. Complete all exercises and problems below.

How to submit

Your submission consists of several steps. You'll need to edit or create the following notebooks in the directory `hw2-files`:

- `exercise_1.ipynb`
- `multivariate-linear-regression.ipynb`

Then follow these directions to submit:

1. **Upload written solutions to Gradescope**. Create a single high-quality pdf with your solutions to the exercises and extra credit, if applicable. The solutions can be typed or written and scanned but the resulting pdf *must* be high quality and easily readable. Upload the pdf to Gradescope.

2. **Upload code listing to Gradescope**. Make sure you set your path to include the anaconda binary directory. On the lab computers, this is the correct command:

   ```
   $ export PATH=/anaconda/bin:$PATH
   ```

   When you are done editing and ready to submit your code listing, run the `pprint_hw2.py` script from the `hw2-files` directory:

   ```
   $ python pprint_hw2.py
   ```

   This will create a new file called `hw2-code.pdf` with a listing of all of your code and results. Open the pdf to make sure it is correct and includes all of your code and plots. You can run this multiple times if you update your code. Upload this to Gradescope.

3. **Submit a single zip file containing source code to Moodle.** Make sure your code is complete and files are included in the directory. Also include any auxiliary data or code files you created that are needed to run your code.

   Rename your code directory from `hw2-files` to `hw2-<your last name>` and zip it:

   ```
   $ mv hw2-files hw2-sheldon
   $ zip -r hw2-sheldon.zip hw2-sheldon
   ```

   Submit the single zip file to Moodle.

Exercises (2 points unless otherwise indicated)

The purpose of these exercises is to practice manipulating matrices and vectors in Python and the basics of matrix multiplication, matrix-vector multiplication, and algebraic rules using matrices and vectors. These are foundational skills for developing machine learning algorithms.

1. (10 points) Create a jupyter notebook called `exercise_1.ipynb` and write code to do the following.
   - Enter the following matrices and vectors

$$A = \begin{bmatrix} -2 & -3 \\ 1 & 0 \end{bmatrix}, \quad B = \begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix}, \quad \mathbf{x} = \begin{bmatrix} -1 \\ 1 \end{bmatrix}$$

   - Compute $C = A^{-1}$
   - Check that $AC = I$ and $CA = I$
   - Compute $A\mathbf{x}$
   - Compute $A^T A$
   - Compute $A\mathbf{x} - B\mathbf{x}$
   - Compute $\|\mathbf{x}\|$ (use the dot product)
   - Compute $\|A\mathbf{x} - B\mathbf{x}\|$
   - Print the first column of A (do not use a loop — use array "slicing" instead)
   - Assign the vector $\mathbf{x}$ to the first column of B (do not use a loop – use "array slicing" instead)
   - Compute the element-wise product between the first column of A and the second column of A
2. Write the result of the following matrix-matrix multiplication. Your answer should be written in terms of u, v, a, and b.

$$\begin{bmatrix} 3 & -1 \\ 2 & 5 \\ -2 & 2 \end{bmatrix} \cdot \begin{bmatrix} u & a \\ v & b \end{bmatrix}$$

3. Suppose $A \in \mathbb{R}^{2\times 2}, B \in \mathbb{R}^{2\times 4}$. Does the product AB exist? If so, what size is it?
4. Suppose $A \in \mathbb{R}^{3\times 5}, B \in \mathbb{R}^{4\times 1}$. Does the product AB exist? If so, what size is it?
5. Suppose $A \in \mathbb{R}^{3\times 2}, \mathbf{y} \in \mathbb{R}^3$. Is $\mathbf{y}^T A$ a row vector or a column vector?
6. Suppose $A \in \mathbb{R}^{3\times 2}, \mathbf{x} \in \mathbb{R}^2$. Is $A\mathbf{x}$ a row vector or a column vector?
7. (5 points) Suppose $(B\mathbf{x} + \mathbf{y})^T A^T = \mathbf{0}$, where A and B are both invertible $n \times n$ matrices, $\mathbf{x}$ and $\mathbf{y}$ are vectors in $\mathbb{R}^n$, and $\mathbf{0}$ is a vector of all zeros. Use the properties of multiplication, transpose, and inverse (see the linear algebra review notebook) to show that $\mathbf{x} = -B^{-1}\mathbf{y}$. Show your work.

Problems

Problem 1 Polynomial Regression (30 points).

In this problem you will implement methods for multivariate linear regression and use them to solve a polynomial regression problem. The purpose of this problem is:

1. To practice writing "vectorized" versions of algorithms in Python
2. To understand how feature expansion can be used to fit non-linear hypotheses using linear methods
3. To understand feature normalization and its impact on numerical optimization for machine learning.

Open the notebook `multivariate-linear-regression.ipynb` and follow the instructions to complete the problem.

Extra Credit Problems

Either write solutions or put them in designated spot at the bottom of `multivariate-linear-regression.ipynb`

1. (4 points) Show that the hypothesis found by the normal equations will be the same regardless of whether or not you normalize features.[1] (**Hint**: show that there is a bijection between hypotheses using the original features and hypotheses using the normalized features. Use the fact that the normal equations find a hypothesis of minimum cost.)
2. (2 points) The data in Problem 1 was generated from a 4th degree polynomial. Try fitting higher degree polynomials to the same data. Observe what happens as you increase the degree of the fitted polynomial.
3. (2 points) There are some cases where $X^T X$ is *not* invertible. In this case, the normal equations still do the right thing if they are modified to use the Moore-Penrose *pseudoinverse*, which is often denoted $(X^T X)^+$. In Python, you can use `numpy.linalg.pinv` instead of `numpy.linalg.inv`. (This also does the right thing if $X^T X$ is invertible, so it is the preferred way of coding the normal equations.) Here are some things for you to explore:
   - Read about the Moore-Penrose pseudoinverse on Wikipedia.
   - Describe two different properties of a training set that would lead to $X^T X$ not being invertible.
4. (2 points) Find your own data where a straight line does not give a good fit, and fit a polynomial to it.

[1] In this model, normalization helps gradient descent find the best hypothesis more easily, but it doesn't change what the best hypothesis is. In later algorithms we will study, normalization also helps improve the quality of the best hypothesis.