CS 475/675 Machine Learning: Homework 2

Due: Friday, October 08, 2021, 11:59 pm US/Eastern

70 Points Total Version 1.0

Make sure to read from start to finish before beginning the assignment.

1 Homeworks

This homeworks will contain three parts:

1. **Programming:** The goal of this programming assignment is for you fit parameters in a logistic regression model by solving estimating equations using the Newton-Raphson method. Unless otherwise stated, pleased do not use any existing packages or libraries (even if an existing implementation already exists as a package, or even a part of standard Python!). If you are unsure whether an existing function can be used, it is safer to ask on Piazza to get our approval before you use it.

2. **Analytical:** These analytical questions will consider topics from the course. These will include mathematical derivations and analyses. Your answers will be entirely based on written work, i.e. no programming.

3. **Practicum:** In the practicum portion of the assignment, you will try feature engineering. Practicums could involve Python notebooks, applied explorations of topics covered in the class, or programming assignments.

   Click here for the Practicum Google Colab Notebook

The point total for each portion of the homework will be listed in the assignment. Written assignments will be submitted as PDFs. See below for more details about what to submit.

1.1 Collaboration Policy

For this assignment, all three parts the programming, the analytical problems and the practicum is intended to be done with a partner (e.g. teams of two students). However, you may also choose to work on and submit the assignment by yourself. If you choose to work with a partner, you and your partner will make one submission for the two of you on Gradescope (make sure to include your partner's information when you submit). You and your partner will receive the same grade, so please choose your partner carefully.

1.2 What to Submit

For this assignment you will submit the following.

1. **Analytical**. You will submit your analytical solutions to Gradescope. **Your writeup must be compiled from LaTeX and uploaded as a PDF**. The writeup should contain all of the answers to the analytical questions asked in the assignment. Make sure to include your name in the writeup PDF and to use the provided LaTeX template for your answers following the distributed template. You will submit this to the assignment called "Homework 2: Analytical".
2. **Practicum Python Notebook** You will submit your Python notebook as a PDF by going to File → Export via PDF or File → Export via PDF via LaTeX. Once you download the pdf, open the file to ensure that the plots show up. You will submit this to the assignment called "Homework 2: Practicum".
3. **Programming Code** You will submit your code and associated files as a zip file. You will submit this to the assignment called "Homework 2: Code".

1.3 Questions?

Remember to submit questions about the assignment to the appropriate group on Piazza: `piazza.com/jhu/fall2021/cs601475675`.

2 Programming (20 points)

We will be using a real-world finance dataset for this homework. You can find the data on Piazza. You will train your algorithm on the train file and use the development set to test that your algorithm works. The test file contains unlabeled examples that we will use to test your algorithm. It is a very good idea to run on the test data just to make sure your code doesn't crash. You'd be surprised how often this happens.

2.1 Data

Finance is a data rich field that employs numerous statistical methods for modeling and prediction, including the modeling of financial systems and portfolios.1 Our financial task is to predict which Australian credit card applications should be accepted (label 1) or rejected (label 0). Each example represents a credit card application, where all values and attributes have been anonymized for confidentiality. Features are a mix of continuous and discrete attributes and discrete attributes have been binarized.

2.2 Data Formats

The data are provided in what is known as SVM-light format. Each line contains a single example:
0 1:-0.2970 2:0.2092 5:0.3348 9:0.3892 25:0.7532 78:0.7280

The first entry on the line is the label. The label can be an integer (0/1 for binary classification) or a real valued number (for regression.) The classification label of -1 indicates unlabeled. Subsequent entries on the line are features. The entry 25:0.7532 means that feature 25 has value 0.7532. Features are 1-indexed.

Model predictions are saved as one predicted label per line in the same order as the input data. The code that generates these predictions is provided in the library. The script compute accuracy.py can be used to evaluate the accuracy of your predictions for classification:

```
python3 compute_accuracy.py test_file test_predictions_file
```

We provide this script since it is exactly how we will evaluate your output. Make sure that your algorithm is outputting labels as they appear in the input files. If you use a different internal representation of your labels, make sure the output matches what's in the data files. The above script will do this for you, as you'll get low accuracy if you write the wrong labels.

2.3 Python

We will be using Python 3.7.3. We are not using Python 2, and will not accept assignments written for this version. We recommend using a recent release of Python 3.7.x, but any recent Python3 should be fine.

For each assignment, we will tell you which Python libraries you may use. We will do this by providing a requirements.txt file. We strongly recommend using a virtual environment to ensure compliance with the permitted libraries. By strongly, we mean that unless you have a very good reason not to, and you really know what you are doing, you should use a virtual environment. You can also use anaconda environments, which achieve the same goal. The point is that you should ensure you are only using libraries available in the basic Python library or the requirements.txt file.

2.4 Virtual Environments

Virtual environments are easy to set up and let you work within an isolated Python environment. In short, you can create a directory that corresponds to a specific Python version with specific packages, and once you activate that environment, you are shielded from the various Python package versions that may already reside elsewhere on your system. Here is an example:

```
# Create a new virtual environment. python3 -m venv python3-hw2
# Activate the virtual environment. source python3-hw2/bin/activate
# Install packages as specified in requirements.txt
pip install -r requirements.txt
# When you wish to deactivate the virtual environment, returning to your system's setup.
deactivate
```

When we run your code, we will use a virtual environment with only the libraries in requirements.txt. If you use a library not included in this file, your code will fail when we run it, leading to a large loss of points. By setting up a virtual environment, you can ensure that you do not mistakenly include other libraries, which will in turn help ensure that your code runs on our systems. Make sure you are using the correct requirements.txt file from the current assignment. We may add new libraries to the file in subsequent assignments, or even remove a library (less likely). If you are using the wrong assignments requirements.txt file, it may not run when we grade it. For this reason, we suggest creating a new virtual environment for each assignment. It may happen that you find yourself in need of a library not included in the requirements.txt for the assignment. You may request that a library be added by posting to Piazza. This may be useful when there is some helpful functionality in another library that we omitted. However, we are unlikely to include a library if it either solves a major part of the assignment, or includes functionality that isn't really necessary.

In this assignments we will allow you to use numpy.

"Can I use the command import XXX?" For every value of XXX the answer is: if you are using a virtual environment setup with the distributed requirements.txt file, then you can import and use anything in that environment.

2.5 Existing Components

The foundations of the learning framework have been provided for you. You will need to complete this library by filling in code where you see a TODO comment. You are free to make changes to the code as needed provided you do not change the behavior of the command lines described above. We emphasize this point: **do not change the existing command line flags, existing filenames, or algorithm names**. We use these command lines to test your code. If you change their behavior, we cannot test your code.

You should spend some time to familiarize yourself with the code we provide. Here is a short summary to get you started:

1. **data.py** – This contains the load data function, which parses a given data file and returns features and labels. The features are stored as a sparse matrix of floats (and in particular as a scipy.sparse.csr matrix of floats), which has num examples rows and num features columns. The labels are stored as a dense 1-D array of integers with num examples elements.

2. **model.py** – This contains a Model class which you should extend. Models have (in the very least) a fit method, for fitting the model to data, and a predict method, which computes predictions from features. You are free to add other methods as necessary.

3. **main.py** – This is the main testbed to be run from the command line. It takes care of parsing command line arguments, entering train/test mode, saving models/predictions, etc. Once again, do not change the names of existing command- line arguments.

4. **compute accuracy.py** – This is a script which simply compares the true labels from a data file (e.g., speech.dev) to the predictions that were saved by running classify.py (e.g., speech.dev.perceptron.predictions).

All of the TODO comments in this assign is in the models.py document. So, that is the only file that you should be writing your code in.

2.6 How to Run the Provided Framework

The framework operates in two modes: train and test. Both stages are defined in main.py, which has the following arguments:

- **--mode** defines which mode to run it, either train or test
- **--train-data** Required in train mode! A file with labeled training data.
- **--test-data** Required in test mode! A file with labeled (dev) or unlabeled (test) data, to perform inference over.
- **--model-file** Required in both modes! In train mode, this will determine where to save your model after training. In test mode, this will tell the program where to load a pretrained model from, to perform inference.
- **--predictions-file** Required in test mode! Tells the program where to store a model's predictions over the test data.
- **--estimating-equation-index** Required in train mode! A function of the features x, used in estimating equations. Since it is difficult to parse string into a math function, we pre-defined a few functions in main.py and you must implement in fit such that the math function corresponding to this estimating equation index is passed in to fit your function. However, make sure your fit function works with all reasonable function A(X), not just the two example functions we provide. I would encourage you to make a duplicate of main.py and add in your own functions to test your program.
- **--tolerance** Required in train mode! Tolerance used to exit the Newton-Raphson loop.

Note that when we run your code, we will use predetermined values for these, to ensure consistency amongst everyone's implementation.

2.7 Logistic Regression

You will implement fitting of a logistic regression model via solving estimating equations using a data set included with the homework.

Your logistic regression function will take extra arguments, the third is a closure (more on closures below) representing a function A(X), and the fourth is tol, which governs how close successive guesses of the parameters have to be before the Newton- Raphson loop exits. Your logistic regression function will solve, via Newton-Raphson, the estimating equations:

$\sum_{i=1}^{n} A(x_{i1:k})\{x_{i(1:k);w}\} = 0.$

Your Newton-Raphson implemention ought to be able to handle any reasonable function A(X).

2.8 Logistic Regression with L2 Regularization

You will implement fitting of a logistic regression model with Ridge Regression/L2 Regularization:

$\sum_{i=1}^{n} -y_i \log(\widehat{y}_i) + (1 - y_i)\log(1 - \widehat{y}_i) + \lambda \sum_{j=1}^{k} \beta_j^2.$

Please do not use any libraries for fitting in this assignment, and implement fitting directly using matrix manipulation. You may find **numpy.linalg** useful for the matrix inversion and the matrix multiplication.

3 Analytical (35 points)

Please see the accompanying `homework2_template.tex` file for the analytical questions for this assignment. There is space provided in that file for you to type your answers in LaTeXafter each question. **Do not edit the file in any way except to add your answers.** Gradescope assumes that the PDF will exactly match our template except for your solutions.

In addition to completing the analytical questions, your assignment for this homework is to learn LaTeX. All homework writeups must be PDFs compiled from LaTeX. Why learn LaTeX?

1. It is incredibly useful for writing mathematical expressions.
2. It makes references simple.
3. Many academic papers are written in LaTeX.

The list goes on. Additionally, it makes your assignments much easier to read than if they are written by hand or if you complete them in Word.

We realize learning LaTeX can be daunting. Fear not. There are many tutorials on the Web to help you learn. We recommend using `pdflatex`. It's available for nearly every operating system. As the semester progresses, you'll no doubt become more familiar with LaTeX, and even begin to appreciate using it.

Be sure to check out this cool LaTeX tool for finding symbols. It uses machine learning! `http://detexify.kirelabs.org/classify.html`

For each homework analytical we will provide you with a LaTeX template. You **must use the template**. The template contains detailed directions about how to use it.

Please open the template to view the analytical questions.

4 Practicum (35 points)

In this assignment you will be trying out feature engineering on a dataset of your choice. Your dataset can be the finance dataset from this assignment, your dataset from Homework 1, or any other dataset of your choice as long as it is suited for supervised machine learning.

4.1 Working with a dataset

Open the Jupyter notebook `homework2_practicum.ipynb`. You will hand in both the Python notebook, which contains answers to the questions, and the dataset you create.

Click here for the Practicum Google Colab Notebook