CS190I HW3: Logistic Regression with L_2 Regularization

Instructor: William Wang
Lead TA: Jiachen Li
Quarter: Spring 2023

Due: May 23rd 1:00pm PT;
submit runs via Kaggle and code via Gradescope

1 Policy on Collaboration among Students

We follow UCSB's academic integrity policy from UCSB Campus Regulations, Chapter VII: "Student Conduct and Discipline"):

"It is expected that students attending the University of California understand and subscribe to the ideal of academic integrity, and are willing to bear individual responsibility for their work. Any work (written or otherwise) submitted to fulfill an academic requirement must represent a student's original work. Any act of academic dishonesty, such as cheating or plagiarism, will subject a person to University disciplinary action. Using or attempting to use materials, information, study aids, or commercial "research" services not authorized by the instructor of the course constitutes cheating. Representing the words, ideas, or concepts of another person without appropriate attribution is plagiarism. Whenever another person's written work is utilized, whether it be a single phrase or longer, quotation marks must be used and sources cited. Paraphrasing another's work, i.e., borrowing the ideas or concepts and putting them into one's "own" words, must also be acknowledged. Although a person's state of mind and intention will be considered in determining the University response to an act of academic dishonesty, this in no way lessens the responsibility of the student."

More specifically, we follow Stefano Tessaro and William Cohen's policy in this class:

- You cannot copy the code or answers to homework questions or exams from your classmates or from other sources;
- You may discuss course materials and assignments with your classmate, but you cannot write anything down.
- You must write down the answers / code independently.
- The presence or absence of any form of help or collaboration, whether given or received, must be explicitly stated and disclosed in full by all involved, on the first page of their assignment.

Specifically, each assignment solution must start by answering the following questions:

1. Did you receive any help whatsoever from anyone in solving this assignment? Yes / No.
   If you answered 'yes', give full details: _____
   (e.g. "Jane explained to me what is asked in Question 3.4")
2. Did you give any help whatsoever to anyone in solving this assignment? Yes / No.
   If you answered 'yes', give full details: _____
   (e.g. "I pointed Joe to section 2.3 to help him with Question 2".)

Academic dishonesty will be reported to the highest line of command at UCSB. When you are not sure, ask the teaching staff before you do so. Students who engage in plagiarism activities will receive an F grade automatically.

2 Programming Assignment (80%)

Logistic regression (LR) classifier is one of the most powerful discriminative classifiers. It allows one to arbitrarily incorporate any features into the model without pain. It is also easy to train LR classifiers using standard constrained or unconstrained optimization techniques, and by incorporating the regularization component, LR is robust to noise. With such nice property, LR is widely used in numerous academic and industrial applications, such as sentiment analysis, question answering, natural language understanding, spoken language processing, and click-through rate prediction.

2.1 Parameter Estimation

In the standard logistic regression model, we define an example $(\vec{x}, y)$, where $y \in \{0, 1\}$ is the label to be predicted, and $\vec{x}$ is a feature vector. The probability of a single example $(\vec{x}, y)$ is defined as

$$P(Y = y|X = \vec{x}, \vec{w}) = \begin{cases} \frac{1}{1+e^{-\vec{w}\vec{x}}} & \text{if } y = 1 \\ 1 - \frac{1}{1+e^{-\vec{w}\vec{x}}} & \text{if } y = 0. \end{cases}$$

To motivate this formula, note that this similar in form to the other linear classifiers we've looked at, like Naive Bayes and the perceptron — it's just that instead of using an argmax or a sign function on the inner product $\vec{w}\vec{x}$, we're using the easy-to-differentiate logistic function $f(z) = \frac{1}{1+e^{-z}}$, which approximates a step function between 0 and 1.

What you will need to do is look at the gradient of $\log P_{\vec{w}}(y|\vec{x})$ for a single example $(\vec{x}, y)$. The algorithm we will use will be to pick a random example, compute this gradient, and adjust the parameters to take a small step in that direction – so we're using a randomized approximation to the true gradient over all the data. This turns out to be fast and surprisingly effective and algorithmically a lot like the perceptron method. As notation, let p be

$$p \equiv \frac{1}{1 + e^{-\vec{w}\vec{x}}} = \frac{1}{1 + \exp(-\sum_j x^j w^j)}$$

and consider the log probability (which, since it's monotonic with the probability, will have the same maximal values):

$$\log P(Y = y|X = \vec{x}, \vec{w}) = \begin{cases} \log p & \text{if } y = 1 \\ \log(1 - p) & \text{if } y = 0. \end{cases}$$

Let's take the gradient with respect to some parameter value w^j. You will consider the two cases separately for now: using $(\log f)' = \frac{1}{f} f'$, we have

$$\frac{\partial}{\partial w^j} \log P(Y = y|X = \vec{x}, \vec{w}) = \begin{cases} \frac{1}{p} \frac{\partial}{\partial w^j} p & \text{if } y = 1 \\ \frac{1}{1-p}(-\frac{\partial}{\partial w^j} p) & \text{if } y = 0 \end{cases}$$

Now you can get the gradient for p, using $(e^f)' = e^f f'$ and the chain rule:

$$\begin{aligned} \frac{\partial}{\partial w^j} p &= \frac{\partial}{\partial w^j} (1 + \exp(-\sum_j x^j w^j))^{-1} \\ &= (-1)(1 + \exp(-\sum_j x^j w^j))^{-2} \frac{\partial}{\partial w^j} \exp(-\sum_j x^j w^j) \\ &= (-1)(1 + \exp(-\sum_j x^j w^j))^{-2} \exp(-\sum_j x^j w^j)(-x^j) \\ &= \frac{1}{1 + \exp(-\sum_j x^j w^j)} \frac{\exp(-\sum_j x^j w^j)}{1 + \exp(-\sum_j x^j w^j)} x^j \end{aligned}$$

Note that

$$1 - p = \frac{1 + \exp(-\sum_j x^j w^j)}{1 + \exp(-\sum_j x^j w^j)} - \frac{1}{1 + \exp(-\sum_j x^j w^j)} = \frac{\exp(-\sum_j x^j w^j)}{1 + \exp(-\sum_j x^j w^j)},$$

so the final line can be rewritten to give

$$\frac{\partial}{\partial w^j} p = p(1-p)x^j,$$

which we can plug into the cases above and get

$$\frac{\partial}{\partial w^j} \log P(Y = y|X = \vec{x}, \vec{w}) = \begin{cases} \frac{1}{p} p(1-p)x^j = (1-p)x^j & \text{if } y = 1 \\ \frac{1}{1-p}(-1)p(1-p)x^j = -px^j & \text{if } y = 0 \end{cases}$$

Finally, these cases can be combined to give the very simple gradient

$$\frac{\partial}{\partial w^j} \log P(Y = y|X = \vec{x}, \vec{w}) = (y-p)x^j.$$

So, taking a small step in this direction would be to increment $\vec{x}$ as follows:

$$\vec{w}^{(t+1)} = \vec{w}^{(t)} + \alpha(y-p)\vec{x}$$

where α is the learning rate. Compare this to the perceptron update rule: it's not very different. Note that the reason we are providing these intermediate steps here is for you to better understand the algorithm. For your implementation, you do not need the detailed derivation.

2.2 L_2 Regularization

Logistic regression tends to overfit when there are many rare features. One fix is to penalize large values of w^j, by optimizing, instead of just optimizing LCL (log-conditional likelihood) in training set (size I, index by i):

$$LCL = \sum_i^I (y_i \log p + (1-y_i)\log(1-p)),$$

we can add a L_2 norm:

$$LCL - \lambda||\vec{w}||^2.$$

Here λ controls how much weight to give to the penalty term. Then instead of

$$\frac{\partial}{\partial w^j} \log P(Y = y|X = \vec{x}, \vec{w}) = (y-p)x^j,$$

we have

$$\frac{\partial}{\partial w^j} \log P(Y = y|X = \vec{x}, \vec{w}) - \lambda \sum_{j=1}^{d} (w^j)^2 = (y-p)x^j - 2\lambda w^j,$$

and the update of $\vec{w}$ becomes

$$\vec{w}^{(t+1)} = \vec{w}^{(t)} + \alpha((y-p)\vec{x} - 2\lambda\vec{w}^{(t)}).$$

2.3 Numerical Optimization

In this homework, you will need to implement your own numerical optimizer using the standard **stochastic gradient descent (SGD)** algorithm. Here is the algorithm:

Initialize the weight vector $\vec{w}$ and the learning rate α;
while *objective function not converged* **do**
  Randomly shuffle the order of the examples in the training set;
  for *each example in the training set* **do**
    $\vec{w} = \vec{w} + \alpha((y-p)\vec{x} - 2\lambda\vec{w})$;
  end
end

To determine when to stop this optimization or the convergence, you need to keep track of the function value in $LCL - \lambda||\vec{w}||^2$. If the change in the function value is less than a small number (e.g. 1.0×10^{-3} or 1.0×10^{-4}), it is likely that your algorithm has found the global optimum.

Note that for this homework, to ensure the convergence, you probably need to use an adaptive learning rate. First you need to set up an initial learning rate α before training (you can try multiple values, e.g. 0.1, 0.01, 0.001, ...). Besides, decay the learning rate with a coefficient (called *learning rate decay*) during training as follows:

$$\alpha = learning\_rate\_decay \times \alpha.$$

Try to monitor the training process to choose the proper learning rate and learning rate decay.

2.4 Making Prediction

In the testing time for binary logistic regression, assuming $\vec{x}$ is a feature vector that you see in the testing time, you can calculate:

$$P(Y = y|X = \vec{x}, \vec{w}) = \begin{cases} \frac{1}{1+e^{-\vec{w}\vec{x}}} & \text{if } y = 1 \\ 1 - \frac{1}{1+e^{-\vec{w}\vec{x}}} & \text{if } y = 0. \end{cases}$$

2.5 Data Set

The columns in the CSV file are labeled as follows:

`id` The document id in the training or test set (positive integer).

`feature_0, feature_1, ..., feature_2999` denoting the feature vector.

`label` **(train.csv only)** The label denoting whether a review is positive or negative. The task is called *sentiment analysis*. There are two different kinds of documents:

- **positive** (labeled with 1), containing positive reviews.
- **negative** (labeled with 0), containing negative reviews.

Your goal is to classify the positive/negative reviews using the logistic regression code you have written. Your prediction output $y \in \{0, 1\}$. Note that this dataset and this task is much more difficult than the dataset and the problem in previous homework. An accuracy of around $70-80\%$ is a very reasonable performance.

We provide `train.csv` and `test_noans.csv`. Your task is to train each of the above classifiers using `train.csv` and produce two `test_answer.csv` files from them. `train` contains all the above columns, while `test_noans` does NOT contain the column, `label`. Your code should generate `test_answer.csv` files which only contain columns `id` and `label` containing your model's guesses.

You can download the two files from the Kaggle competition page[1]. The `Data` section contains the training data [2] and test data[3]. (Details on participating in the competitions in section 4)

2.6 Implementation notes

You need to select the best hyper-parameters based on 10-fold cross-validation (CV) on the training set, including the regularization strength λ, the learning rate α and the learning rate decay. (http://en.wikipedia.org/wiki/Cross-validation_(statistics).) To do this, take the regularization strength as an example, you can define some candidate λs (e.g. $\lambda = 1 \times 10^{-7}, 1 \times 10^{-6}, \ldots, 1$), run CV using each candidate λ, and then compare the averaged results of each fold in CV. You select the λ that gives you the best performance on training set, and use that best hyper-parameter, retrain the model on the entire training set, and make prediction on the test set. In other words, you have to choose the best hyper-parameters using CV offline and set them up in your submitted code, so that your submitted code will simply run once on the training data and then make the predictions on the test data. Your code must be written in Python.

[1] https://www.kaggle.com/competitions/ucsb-cs190i-hw3-logistic-regression-with-l2-reg

[2] https://www.kaggle.com/competitions/ucsb-cs190i-hw3-logistic-regression-with-l2-reg/data?select=train.csv

[3] https://www.kaggle.com/competitions/ucsb-cs190i-hw3-logistic-regression-with-l2-reg/data?select=test_noans.csv

2.7 Evaluation Criteria

We will then compare the prediction file you submit with the ground truth file and use F1-score to evaluate your results. The credit you receive on the this part of the assignment will be assigned as follows:

- 85/80 points: $F_1 >= 0.74$.
- 80/80 points: $0.74 > F_1 >= 0.73$.
- 75/80 points: $0.73 > F_1 >= 0.72$.
- 70/80 points: $0.72 > F_1 >= 0.71$.
- 65/80 points: $0.71 > F_1 >= 0.70$.
- 60/80 points: $0.70 > F_1 >= 0.65$.
- 40/80 points: $0.65 > F_1 >= 0.60$.
- 20/80 points: $0.60 > F_1 >= 0.55$.
- 0/80 points: $F_1 < 0.55$.

3 Report (20%)

You should submit a report to Gradescope alongside your Python code that produces `test_answer.csv` file using the abovementioned techniques. The report should solve the following question:

Investigate the effects of regularization on your training data. For this purpose, try $\lambda = \{0, 1 \times 10^{-7}, 1 \times 10^{-6}, 1 \times 10^{-5}, 1 \times 10^{-4}, 1 \times 10^{-3}, 1 \times 10^{-2}, 1 \times 10^{-1}, 1\}$ and report the accuracy of cross-validation on the training set. Plot the accuracy as a function of the λs (the x-axis should be λ and the y-axis should be "accuracy"). **You should write a paragraph interpreting your results, with explanations for why you think the results look the way they do.**

4 Submission

4.1 Writing and submitting your code and report to Gradescope

Write your code in Python. You should use the `numpy` package to implement your solutions, and we suggest using the package `pandas` to load the datasets.

Your code should read the files `train.csv` and `test_noans.csv` and produce output file `test_ans.csv` using both the naive Bayes and voted perceptron techniques described above. You will submit your runs using Kaggle (**4.2**). You will submit your code and report using Gradescope.

Submission format You will submit two files to Gradescope: `run.py` and `report.pdf`. The `run.py` file should contain all code necessary to produce your output file `test_ans.csv`.

We will run your code to check for accuracy and plagiarism check your code, example run outputs, and report. The only non-default packages you are allowed to use in this submission are `numpy` and `pandas`.

Gradescope links:

1. `run.py` : gradescope.com/courses/527963/assignments/2883688/
2. `report.pdf` : gradescope.com/courses/527963/assignments/2883828/

Additionally, you **must submit** your **test_ans.csv** file to the Kaggle competition as described below.

4.2 Submitting your runs to Kaggle for result evaluation

Kaggle is a popular data science competition website for both learners and companies looking to hire. Some high-profile Kaggle competitions include cash prizes, and being a top-ranked Kaggler can even boost for your resume! Please sign up for an account at kaggle.com, as we will be partially grading our coding assignments using Kaggle.

To submit your result, go to kaggle.com/t/e88c2239e1ea436da62cf761462198d6. If you are making **late submissions after 5/23 only**, use these links instead to submit your runs: kaggle.com/t/05b784f9dafe40c686ce9d8f22e9c535.

IMPORTANT: Please submit your runs with YOUR UCSB NETID as your team name! This is critical so we can assign your grade based on your submission.