CS190I Introduction to NLP Assignment 3: Logistic Regression with L_2 Regularization

Due: 11/15, 23:59pm PT

Late submission due: 11/19, 23:59pm PT,
via late submission page with 50% discount of credits.

Instructor: William Wang, TA: Jing Qian

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 $(\mathbf{x}, y)$, where $y \in \{0, 1\}$ is the label to be predicted, and $\mathbf{x}$ is a feature vector. The probability of a single example $(\mathbf{x}, y)$ is defined as

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

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 $\mathbf{wx}$, 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 $\log P_{\mathbf{w}}(y|\mathbf{x})$ for a single example $(\mathbf{x}, y)$. 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

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

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

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

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

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

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

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

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

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

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

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

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:

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

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

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

we have

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

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

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

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 $\mathbf{w}$ 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**
    $\mathbf{w} = \mathbf{w} + \alpha((y-p)\mathbf{x} - 2\lambda\mathbf{w})$;
  end
end

To determine when to stop this optimization or the convergence, you need to keep track of the function value in $LCL - \lambda||\mathbf{w}||^2$. 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 $\mathbf{x}$ is a feature vector that you see in the testing time, you can calculate:

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

2.5 Data Set

For this assignment, you should download the training data from https://www.dropbox.com/s/ibsmps152i8vz0w/hw3_data.zip?dl=0. There are two csv files included. Similar to the previous homework, "Xtrain.csv" stores the training feature matrix. "Ytrain.csv" stores the testing labels.

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.

2.6 Implementation & Submission

For this assignment, we publish a CodaLab competition (https://competitions.codalab.org/competitions/20466?secret_key=36699e8d-781e-4a1a-ba05-edb78134eb11), where you can submit your code and get evaluation scores that will be listed on a leader-board. If you are not familiar with CodaLab Competitions, check out here: participating in a competition.

Remember to register a CodaLab Competitions account using your umail account, so that the username will be your UCSBNetID. After that, log into CodaLab Compectitions and set up your team name (whatever nickname you like). To protect your privacy, only the team names will be shown on the leader-board and your usernames will be anonymous. After your submission finishes running, please choose to submit it to the leader-board. Note that here team name is equivalent to your nickname, and it is still an independent homework assignment. You must implement the two algorithms according to the instructions above. You must avoid calling APIs from existing machine learning toolkits such as scikit-learn. Using numpy is allowed.

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. $\lambda = 1\times10^{-7}, 1\times10^{-6}, ..., 1$), 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.

Submission format The final submission format should be:
logistic_regression.zip

- run.py
- other python scripts you wrote
- report.pdf (see Sec. 3 for more details)

Note that to create a valid submission, zip all the necessary files with 'zip zipfilename file1 file2 ...' starting from this directory. **DO NOT zip the directory itself, just its contents. Only zip your code and report and DO NOT include any other unnecessary files.**

Here is a sample 'run.py' file: https://www.dropbox.com/s/a2ahv2qh7solhhn/run.py?dl=0. You must submit the 'run.py' file in the exactly same format. You will have a total of 30 possible submissions.

Computational restrictions For both phases, the prescribed "time budget" is 300 seconds, including both the running time of your code and the evaluation time of the predicted results. In other words, the running time of your program must be less than 300 seconds. **Emphasize again, DO NOT run cross-validation in your submitted code.**

2.7 Evaluation Criteria

As illustrated in Section 2.6, the prediction file generated by your code should be in the exactly same format of the file 'Ytrain.csv'. We will then compare your predictions with the ground truth labels and use a weighted combination of accuracy and F-measure to evaluate your results. So your final score for each submission

$$final\_score = 50 \times accuracy + 50 \times F\_measure.$$

Even though your code is automatically evaluated on CodaLab Competitions, we will download your latest submissions and run a Plagiarism detection program on them. So please be honest.

2.8 Late Submission Policy

Considering that you have a total of four late days for your homework, we create a copy of the CodaLab competition for you to submit you code after the normal deadline. Here is the CodaLab Competition for Late Submission. Note that:

- This competition ends four days later than the normal deadline, which is 11/19, 23:59pm PT.
- The time of your last submission will be used to count the late days you have used.
- Your late days are counted by days, not hours or even minutes. For example, even though you are only late for 1 second, that would be counted as 1 day.
- If you have already run out of four late days, then your late submissions would be penalized with 50% discount of credits.
- Submissions later than 11/19 23:59pm PT are not acceptable by any means.

3 Report (20%)

In addition to the code submission on CodaLab Competitions, you are also supposed to write a report and include it in the submission zip file (at least in the latest submission zip file). 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 (x-axis should be "lambda" and y-axis should be "accuracy").