MiniProject 1: Machine Learning 101

COMP 551, McGill University
Professor: William L. Hamilton

January 17, 2019

Please read this entire document before beginning the assignment.

Preamble

- This mini-project is **due on Jan 31st at 11:59pm.** Late work will be automatically subject to a 20% penalty, and can be submitted up to 5 days after the deadline. No submissions will accepted after this 5 day period.
- This mini-project is to completed in groups of three. There are three tasks outlined below which offer one possible division of labour, but how you split the work is up to you. All members of a group will recieve the same grade. It is not expected that all team members will contribute equally to all components. However every team member should make integral contributions to the project.
- You will submit your assignment on MyCourses as a group. You must register your group on MyCourses and any group member can submit. See MyCourses for details.
- You are free to use libraries with general utilities, such as matplotlib, numpy and scipy for python. **However, you should implement the algorithms yourself, which means you should not use pre-existing implementations of the algorithms as found in SciKit learn, Tensorflow, etc.**

Background

In this miniproject you will use linear regression to predict the popularity of comments on Reddit. Reddit (`www.reddit.com`) is a popular website where users can form interest-based communities, post content (e.g., images, links to news articles), and participate in thread-based discussions.

We have provided you with a (somewhat) preprocessed dataset of Reddit comments from the community `r/AskReddit`, one of the larger, general question-answering forums on Reddit. Please download this dataset and some starter code for loading the data at: `https://www.cs.mcgill.ca/~wlh/comp551/files/proj1_materials.zip`.

Warning: This is messy, real-world Reddit data. It will almost certainly contain statistical oddities, strange phenomena, and (possibly) foul language.

Task 1: Construct the dataset and extract features

Your first task is to construct training, validation, and testing datasets out of the Reddit data dumps we provide. As is noted in the starter code, we provide the data as a (Python) list of data points. Each data point is a (Python) dictionary containing the following information:

- **popularity_score**: This is a score indicating how popular the comment is. *You don't need to worry about how exactly this was computed. The key point is that it measures the popularity of the comment and is what we are trying to predict (i.e., the target variable).*
- **children**: This counts how many replies this comment recieved.
- **controversiality**: This is a metric of how "controversial" a comment is. It is a proprietary metric computed by Reddit and takes on the values $\{0, 1\}$.
- **is_root**: This is a binary variable indicating whether this comment is the "root" comment of a discussion thread (i.e., if this variable is True, then this comment is not a reply to another comment).
- **text:** This is the raw text of the comment.

Split the data

First, partition your data into train, validation, and test splits. There are 12,000 data points in total. Use the first 10,000 points for training, the next 1,000 for validation, and the final 1,000 for testing. **Important: The dataset has been pre-shuffled; to facilitate grading, please split the data in the order it was given (i.e., use the first the 1st to 10,000th data points in the list we provided for training; the 10,001st to 11,000th for validation; etc.)**

Compute some text features

The children, is_root, and controversality features are straightforward to encode using techniques discussed in the lectures (i.e., you can just use the numeric variables as is and encode the binary is_root variable using 0 and 1). The text information on the other hand will require some work.

Pre-processing the text. Processing text data can be tricky, but in this miniproject we are going to keep it simple. First off, with regards to simple pre-processing: we are going to *lower-case all the text* (i.e., ignore capitalization) and we are *going to simply split the input text according to whitespace tokens to define different "words"*. (Yes, this means that punctuation marks will count as words, but that's okay!). **Hint:** You might want to write a simple Python function that processes a string of text by lower-casing and splitting on whitespace...

Word count features. Now, presuming you have a handle on pre-processing, we are going to compute some word count occurrence features. Find the 160 most frequently occurring words over all comments in the training data. Let w_i denote the ith most-frequent word. Now, for every comment make a 160-dimensional feature vector, $\mathbf{x}_{\text{counts}}$, where each entry, $\mathbf{x}_{\text{counts}}[i]$, is equal to the number of times word w_i occurs in that comment. For instance, suppose "the" is the most frequent word in the training data and that "the" occurs two times in a particular comment; then for this comment, we would have that the first entry in the $\mathbf{x}_{\text{counts}}$ feature is equal to two (i.e., $\mathbf{x}_{\text{counts}}[0] = 2$).

Hint: For the computation of the text features, it might be advantageous to wrap all of these preprocessing steps into a generic function (e.g., with input: list of dictionaries, output: matrix $\mathbf{X}$ and vector $\mathbf{y}$) so that this function can easily be reapplied on the training, validation, and test data.

Brainstorm some other features

In addition to the text features above, you must also come up with at least two more features. These can be based on the text data, transformations of the other numeric features, or interaction terms. However, **you will need to demonstrate that these features improve your model (see Task 3)**.

Task 2: Implement linear regression

You must implement linear regression in two ways:

1. Using the closed-form solution $\hat{\mathbf{w}} = (\mathbf{X}^\top \mathbf{X})^{-1}\mathbf{X}^\top \mathbf{y}$
2. Using gradient descent (see Algorithm 1). Note that we suggest using a decaying learning rate in Algorithm 1, but you are free to set the learning rate in any way you see fit. And you can also initialize the parameter vector $\mathbf{w}_0$ in any way you see fit (e.g., randomly or all zeros).

Hint: If your model is not working, you might want to double check that you included the bias term.

Hint: You don't need to use the full dataset while debugging. You can just use a small subset.

Input : Data matrix $\mathbf{X}$, targets $\mathbf{y}$, initial weights $\mathbf{w}_0$, hyperparameters: β, η_0, ϵ
Output: Estimated weights $\hat{\mathbf{w}}$
i = 1;
do
  $\alpha = \frac{\eta_0}{1+\beta i}$;
  $\mathbf{w}_i = \mathbf{w}_{i-1} - 2\alpha\left(\mathbf{X}^\top \mathbf{X}\mathbf{w}_{i-1} - \mathbf{X}^\top \mathbf{y}\right)$;
while $\|\mathbf{w}_i - \mathbf{w}_{i-1}\|_2 > \epsilon$;

Algorithm 1: Gradient descent for linear regression with decaying learning rate α. The η_0 hyperparameter is the initial learning rate and the β hyperparameter controls the speed of the decay.

Task 3: Run experiments

The goal of this project is to have you explore linear regression and compare different features, learning rates, etc. You are welcome to perform any experiments you see fit (e.g., to justify your choice of learning rate), but at a minimum you must complete the following experiments. In all these experiments you should use the mean-squared error on the training and validation sets to report performance. **Note: (nearly) all these experiments should be on the validation set! Only report your test set performance for your final, best-performing model.**

1. Compare the runtime, stability, and performance of the closed-form linear regression and gradient descent approaches. For these experiments, it is fine to ignore the text features and only use the 3 simple features we provide. For gradient descent make sure you try out different learning rates and initializations! (And note that the learning rate might need to be very small...)
2. Using either the closed-form approach or gradient descent, compare a model with no text features, a model that uses only the top-60 words, and a model that uses the full 160 word occurrence features. Are any of these models underfitting or overfitting?
3. Using either the closed-form approach or gradient descent, demonstrate the the two new features you proposed improve performance on the validation set.
4. Run your best-performing model on the test set.

Deliverables

You must submit three separate files to MyCourses (**using the exact filenames and file types outlined below**):

1. **code.zip**: Your data processing and linear regression code (as some combination of .py and .ipynb files).
2. **words.txt**: A text file with the top-160 most frequent words in the training data, ordered in descending order by frequency/count (i.e., a text file with the top-160 most frequent words that you used to construct the word count features).
3. **writeup.pdf**: Your (max 4-page) project write-up as a pdf (details below).

Project write-up

Your team must submit a project write-up that is a maximum of four pages (single-spaced, 11pt font or larger; an extra page for references/bibliographical content can be used). We highly recommend that students use LaTeX to complete their write-ups. **This first mini-project report has relatively strict requirements, but as the course progresses your project write-ups will become more and more open-ended**. You have some flexibility in how you report your results, but you must adhere to the following structure and minimum requirements:

Abstract (100-250 words) Summarize the project task and your most important findings. For example, include sentences like "In this project we investigated the performance of linear regression models for predicting comment popularity on Reddit", "We found that ?? features improved performance and that the gradient descent approach was faster/slower than the closed-form approach.

Introduction (5+ sentences) Summarize the project task, the dataset, and your most important findings. This should be similar to the abstract but more detailed. You can also include extra background information and citations to relevant work (e.g., other papers analyzing Reddit or predicting post/comment popularity).

Dataset (5+ sentences) Very briefly describe the dataset (e.g., size, train/val/test split proportions) and how you extracted the text features. Describe the new features you come up with in detail. Highlight any possible ethical concerns that might arise when working with a public social media dataset of this variety. **Note:** You do not need to explicitly verify that the data satisfies the i.i.d. assumption (or any of the other formal assumptions for linear regression).

Results (7+ sentences, possibly with figures or tables) Describe the results of all the experiments mentioned in Task 3 (at a minimum) as well as any other interesting results you find. At a minimum you must report:

1. A runtime comparison between the gradient descent and closed-form solution approaches.
2. The train and validation mean-squared errors for a model with no text features, a model that uses the text features but is restricted to only using the top-60 words, and a model that uses the full 160 word occurrence features.
3. The train, validation, and test performance of a model that incorporates your new features.

Discussion and Conclusion (5+ sentences) Summarize the key takeaways from the project and possibly directions for future investigation.

Statement of Contributions (1-3 sentences) State the breakdown of the workload across the team members.

Evaluation

The mini-project is out of 100 points, and the evaluation breakdown is as follows:

- Completeness (40 points)
  - Did you submit all the materials?
  - Did you run all the required experiments?
  - Did you follow the guidelines for the project writeup?
- Correctness (40 points)
  - Are your closed-form and gradient descent implementations correct?
  - Are your reported mean-squared errors close to our reference solutions?
  - Do your proposed features actually improve performance?
  - Do you observe the correct trends in the experiments (e.g., comparing a model with no text features to a model with text features)?
- Originality / creativity (10 points)
  - Did you go beyond the bare minimum requirements for the experiments?
  - Did you report the experimental results in a thoughtful way (e.g., using figures or tables)?
  - **Note:** Simply adding in a random figure will not give you extra points in this section. The purpose of adding a figure or table should be to communicate specific high-level results and minimize the amount of effort it takes for the reader to understand what you are trying to say.
- Writing quality (10 points)
  - Is your report clear and free of grammatical errors and typos?
  - Did you go beyond the bare minimum requirements for the write-up (e.g., by including a discussion of related work in the introduction)?

Final remarks

You are expected to display initiative, creativity, scientific rigour, critical thinking, and good communication skills. You don't need to restrict yourself to the requirements listed above - feel free to go beyond, and explore further.

You can discuss methods and technical issues with members of other teams, but you cannot share any code or data with other teams. Any team found to cheat (e.g. use external information, use resources without proper references) on either the code, predictions or written report will receive a score of 0 for all components of the project.