MiniProject 2: Classification of Textual Data

COMP 551, Winter 2021, McGill University
Contact TAs: Tianzi Yang and Haque Ishfaq

Please read this entire document before beginning the assignment.

Preamble

- This mini-project is **due on February 28th at 11:59pm (EST, Montreal Time).** There is a penalty of 2^k percent penalty for k days of delay, which means your grade will be scaled to be out of $100-2^k$. No submission will be accepted after 6 days of delay.
- This mini-project is to be completed in groups of three. All members of a group will receive the same grade except when a group member is not responding or contributing to the project. If this is the case and there are major conflicts, please reach out to the group TA for help and flag this in the submitted report. Please note that it is not expected that all team members will contribute equally. However every team member should make integral contributions to the project, be aware of the content of the submission and learn the full solution submitted.
- 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.
- We recommend to use **Overleaf** for writing your report and **Google colab** for coding and running the experiments. The latter also gives access to the required computational resources. Both platforms enable remote collaborations.
- You should use Python for this and the following mini-projects. You are free to use libraries with general utilities, such as matplotlib, numpy and scipy for Python, unless stated otherwise in the description of the task. In particular, in most cases you should implement the models and evaluation functions yourself, which means you should not use pre-existing implementations of the algorithms or functions as found in SciKit learn, and other packages. The description will specify this in a per case basis.

Background

In this miniproject you will **implement naive Bayes and K-fold cross-validation from scratch**, while **using logistic regression** [1] **from scikit-learn package** (or optionally implemented from scratch) – and compare these two algorithms on two distinct textual datasets. The goal is to gain experience implementing these algorithms from scratch and to get hands-on experience comparing performance of different models.

[1] `https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html`

Task 1: Acquire and preprocess the data

Your first task is to acquire the data and clean it (if necessary). To turn the text data into numerical feature, you should use the bags of words representation using the scikit-learn function `CountVectorizer`[2] for more details on the function. Please use these online resources as guidelines and avoid direct copy and pasting from them for your project, make sure to understand each line and try to write it yourself when using these guides.

We will use two datasets in this project, outlined below.

- 20 news group dataset. Use the default train subset (subset='train', and **remove=(['headers', 'footers', 'quotes'**]) in sklearn.datasets) to train the models and report the final performance on the test subset. Note: you need to start with the text data and convert text to feature vectors. Please refer to `https://scikit-learn.org/stable/tutorial/text_analytics/working_with_text_data.html` for a tutorial on the steps needed for this. You can refer to this tutorial for further tips on how to do data-processing with text data.
- IMDB Reviews: `http://ai.stanford.edu/~amaas/data/sentiment/` Here, you need to use only reviews in the train folder for training and report the performance from the test folder. You need to work with the text documents to build your own features and ignore the pre-formatted feature files.

You are free to use any Python libraries you like to extract features and preprocess the data.

Task 2: Implement Naive Bayes and k-fold cross validation

You are free to implement these models as you see fit, but you should follow the equations that are presented in the lecture slides, and you must implement the models from scratch (i.e., you cannot use SciKit Learn or any other pre-existing implementations of these methods).

In particular, your two main tasks in the part are to:

1. Implement naive Bayes, using the appropriate type of likelihood for features.
2. Implementing k-fold cross validation.

You are free to implement these models in any way you want, but you must use Python and you must implement the models from scratch (i.e., you cannot use SciKit Learn or similar libraries). Using the numpy package, however, is allowed and encouraged. Regarding the implementation, we recommend the following approach:

- Implement naive Bayes model as a Python class. You should use the constructor for the class to initialize the model parameters as attributes, as well as to define other important properties of the model.
- Your model class should have (at least) two functions:
  - Define a `fit` function, which takes the training data (i.e., $\mathbf{X}$ and $\mathbf{y}$)—as well as other hyperparameters (e.g., the learning rate and/or number of gradient descent iterations)—as input. This function should train your model by modifying the model parameters.
  - Define a `predict` function, which takes a set of input points (i.e., $\mathbf{X}$) as input and outputs predictions (i.e., $\hat{\mathbf{y}}$) for these points.
- In addition to the model classes, you should also define a functions `evaluate_acc` to evaluate the model accuracy. This function should take the true labels (i.e., $\mathbf{y}$), and target labels (i.e., $\hat{\mathbf{y}}$) as input, and it should output the accuracy score.
- Lastly, you should implement a script to run k-fold cross-validation for hyper-parameter tuning and model selection. Your implementation should have (at least) three parts below:

[2] `https://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.CountVectorizer.html`

- Define a `cross_validation_split` function, which takes training data as input and splits it into k folds. Each fold is then used once as validation while the $k-1$ remaining folds form the training set. (You may shuffle the data before splitting)
- Define function `kfoldCV`, which takes train/validation sets generated above and a given model as input. The function iterate through each train/validation set and returns the average result of each fold.
- Repeat functions above to the given model with different hyper-parameter settings, select the best hyper-parameter by the best average k-fold result.

You are free to use any Python libraries you like to tune the hyper-parameters, for example see `https://scikit-learn.org/stable/auto_examples/model_selection/plot_randomized_search.html#sphx-glr-auto-examples-model-selection-plot-randomized-search-py`.

Task 3: Run experiments

The goal of this project is to have you explore linear classification and compare different features and models. *Use 5-fold cross validation to estimate performance in all of the experiments. Evaluate the performance using accuracy.* You are welcome to perform any experiments and analyses you see fit (e.g., to compare different features), **but at a minimum you must complete the following experiments in the order stated below**:

1. We expect you to conduct multiclass classification on the 20 news group dataset, and binary classification on the IMDb Reviews.
2. In a single table, compare and report the performance of the performance of naive Bayes and logistic regression on each of the two datasets (with their best hyperparameters), and highlight the winner for each dataset and overall. You could find full hyper-parameters for logistic regression here. [3]
3. Further, with a plot, compare the accuracy of the two models as a function of the size of dataset (by controlling the training size). For example, you can randomly select $20\%, 40\%, 60\%$ and 80% of the available training data and train your model on this subset. Now, compare the performance of corresponding models and highlight the best. [4].

Note: The above experiments are the minimum requirements that you must complete; however, this project is open-ended. For this part, you might implement logistic regression from scratch (and try different learning rates, investigate different stopping criteria for the gradient descent), try linear regression for predicting ratings in the IMDB data, try different text embedding methods as alternatives to bag of words. You are also welcome and encouraged to try any other model covered in the class, and you are free to implement them yourself or use any Python library that has their implementation, e.g. the from the scikit-learn package. Of course, you do not need to do all of these things, but look at them as suggestions and try to demonstrate curiosity, creativity, rigour, and an understanding of the course material in how you run your chosen experiments and how you report on them in your write-up.

Deliverables

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

1. **code.zip**: Your data processing, classification and evaluation code (as some combination of .py and .ipynb files).
2. **writeup.pdf**: Your (max 5-page) project write-up as a pdf (details below).

[3] Logistic Regression: https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html

[4] As an example, see Figure 1 in Ng AY, Jordan MI. On discriminative vs. generative classifiers: A comparison of logistic regression and naive bayes. In Advances in neural information processing systems 2002 (pp. 841-848).

Project write-up

Your team must submit a project write-up that is a maximum of five pages (single-spaced, 11pt font or larger; minimum 0.5 inch margins, 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 classification models on two benchmark datasets", "We found that the logistic regression approach was achieved worse/better accuracy than naive Bayes and was significantly faster/slower to train."

Introduction (5+ sentences) Summarize the project task, the two datasest, and your most important findings. This should be similar to the abstract but more detailed. You should include background information and citations to relevant work (e.g., other papers analyzing these datasets).

Datasets (5+ sentences) Very briefly describe the and how you processed them. Describe the new features you come up with in detail. Present the exploratory analysis you have done to understand the data, e.g. class distribution.

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 comparison of the accuracy of naive Bayes and logistic regression on both datasets.
2. Results demonstrating that the feature subset and/or new features you used improved performance.

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 (20 points)
  - Did you submit all the materials?
  - Did you run all the required experiments?
  - Did you follow the guidelines for the project write-up?
- Correctness (40 points)
  - Are your models implemented correctly?
  - Are your reported accuracies close to the reference solutions?
  - Do your proposed features actually improve performance, or do you adequately demonstrate that it was not possible to improve performance?
  - Do you observe the correct trends in the experiments (e.g., comparing learning rates)?
- Writing quality (25 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)?
- Do you effectively present numerical results (e.g., via tables or figures)?

- Originality / creativity (15 points)
  - Did you go beyond the bare minimum requirements for the experiments?
  - **Note:** Simply adding in a random new experiment will not guarantee a high grade on this section! You should be thoughtful and organized in your report.

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.**