Project: Using a linear classifier to train the LEDGAR set

Last updated: February 27, 2023
Goal

- A basic understanding of how linear text classification works
- Get familiar with the LibMultiLabel linear API
Outline

1. Linear Text Classification
2. Project Contents
Outline

1. Linear Text Classification
2. Project Contents
Text classification is the task of assigning a class to a given document. It has been applied in many fields for automatically categorizing news, legal documents, and electronic medical records, etc.

You might be curious about how it works. Let’s start with an example
Example 1

- **Text**: During the Term of Executive’s Agreement, the Executive shall be entitled to be paid vacation in accordance with the most favorable plans ...
- **Class**: 93 (Vacations)
- The text snippet is the regulation of the paid holidays, so it belongs to class **Vacations**
To perform text classification, we will start by vectorizing text to the feature vector.

One effective way is **Bag of Words (BOW)** which converts text to a column vector of word counts.

So going back to the example, the vocabulary of the given text is:

```
accordance, agreement, be, during, entitled, executive, favorable, in, most, of, paid, plans, shall, term, the, to, vacation, with
```
The corresponding BOW representation:
\[ \mathbf{x} = [1, 1, 2, 1, 1, 2, 1, 1, 1, 1, 1, 1, 1, 1, 3, 1, 1, 1] \]
where 3 is the number of word “the”

Another way to vectorize text is **TF-IDF (Term Frequency-Inverse Document Frequency)**
We’re not going deep in this class. You can find more information [here](#)
After transforming a text to a feature vector, we can predict a class (say Vacations) by the linear techniques taught in the first lesson.

But we only covered binary classification in class.

There are several methods for multi-class classification, including one-vs-rest, one-vs-one, etc.
The following describes how one-vs-rest works:

- Assume $\ell$ is the number of classes. For class $i = 1 \ldots \ell$, we train a binary classifier $\mathbf{w}_i$ using instances belonging to class $i$ as positive instances, and the others as negative instances.
- After obtaining classifiers $\mathbf{w}_1, \ldots, \mathbf{w}_\ell$, given an instance $\mathbf{x}$, we predict the class $\hat{y}$ with the largest decision value, i.e.,

$$
\hat{y} \in \arg \max_{1 \leq i \leq \ell} \mathbf{w}_i^T \mathbf{x}.
$$
An example: assume $\ell = 3$ and we have $x_1, x_2$: class 1, $x_3$: class 2 and $x_4$: class 3.

- Train $w_1$ using
  $\{(x_1, +1), (x_2, +1), (x_3, -1), (x_4, -1)\}$

- Train $w_2$ using
  $\{(x_1, -1), (x_2, -1), (x_3, +1), (x_4, -1)\}$

- Train $w_3$ using
  $\{(x_1, -1), (x_2, -1), (x_3, -1), (x_4, +1)\}$

Given unknown instance $x$, predict the class with the largest decision value
Outline

1. Linear Text Classification
2. Project Contents
Project Setup I

- Your codes and experiments are required to run on /tmp2 of 217 workstations (linux[1-15]) so that we can check your codes if needed.

- Please set up an virtual environment as the following:

  ```bash
  virtualenv -p /usr/bin/python3 /tmp2/$USER/venv
  ```

- The downloaded pip cache is stored in the home directory, which takes up large space. You can create symbolic links to /tmp2.
Download the LEDGAR (LexGLUE) set (ledgar_lexglue_raw_texts_*.txt.bz2) to data

Install LibMultiLabel in the virtual environment
  . venv/bin/activate
  pip3 install libmultilabel
Let’s run linear classification with LibMultiLabel API

The tutorial shows you how to train a linear classifier with customized TF-IDF features

```python
vectorizer = TfidfVectorizer(
    max_features=20000,
    min_df=3)
```

You are going to try feature generation methods to demonstrate what vectorizer/parameters are suitable for the LEDGAR set (see Text feature extraction)
After obtaining the corresponding numerical features for each text, we would like to compare the performance between logistic regression and SVM. In the class, we mentioned that their performance is similar. Please check whether it’s the case for this dataset.

Besides, you’re also going to check whether proper regularization indeed improves the classification results by selecting different $C$’s.
To test different classification methods and parameters, you need to pass different options strings to liblinear. Details can be found in this README page.

LibMultiLabel supports easy hyperparameter selection; see this tutorial about GridSearchCV for reference.
Submission 1

- Write a 2-page report about the experimental results and findings on the following questions:
  - For the LEDGAR set, does changing the parameters of vectorizer improve the performance? For example, what are the pros and cons of reducing the feature size?
  - Are the test performances for logistic regression and SVM similar? If not, could you provide some explanations?
Submission II

Does the optimal regularization parameter $C$ vary based on the choice of max_features in TfidfVectorizer?

Please specify on which /tmp2 you store your code and keep it private before the deadline to avoid plagiarism.

Upload your report in PDF format to NTU COOL before 2023/03/14 23:59