1 Introduction

In this project, you are going to classify whether a review is positive or negative. In this project, you will learn

- how to prepare feature vectors from text data,
- how to use packages for classification, and
- how to train an accurate model.

2 Text Data for Sentiment Analysis

We have 3000 reviews collected from three domains: imdb.com, amazon.com, yelp.com. We put all reviews in a single dataset. Each review consists of a sentence and a sentiment label (1 for positive and 0 for negative) of the sentence. The dataset is given in a single text file, with each line as an instance. Each line is a sentence, followed by a tab character, and then followed by the label. Here is a snippet from the training set:

<table>
<thead>
<tr>
<th>Sentence</th>
<th>Label</th>
</tr>
</thead>
<tbody>
<tr>
<td>Crust is not good.</td>
<td>0</td>
</tr>
<tr>
<td>Best burger ever had!</td>
<td>1</td>
</tr>
</tbody>
</table>

You are suggested to remove all the punctuation, numeric values, and convert upper case to lower case for each example so that the same word will treated in the same way in the data.

In text analysis, we often call an instance as a document. Here a document is actually just a review, but we follow the convention and call a review as a document. To apply classification algorithms to the classification task, we need first convert a document to a vector. There are two approaches, bag-of-words and document embedding. You should use both approaches in this project.

2.1 Bag of words

Indicated by the name, the methods of bag-of-words treats each document as a set of words, and then use the set indicator as the feature vector of the document. Here each word $j$ is a feature. If a document has the word $j$, then its $j$-th feature takes value 1; otherwise its $j$-th feature is 0. In this way, each document is converted to a vector with length of the vocabulary size.
Consider the illustration from class with positive examples: what a nice day and a green cat chased a green dog and negative example green umbrella a nice day. Then the feature table is given by

<table>
<thead>
<tr>
<th>word</th>
<th>a</th>
<th>nice</th>
<th>day</th>
<th>green</th>
<th>cat</th>
<th>chased</th>
<th>dog</th>
<th>umbrella</th>
<th>Class</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>+</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>+</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>-</td>
</tr>
</tbody>
</table>

If you want to keep information about the count of a word $j$, you can word count, instead of binary values, as the feature value. For example, “what a nice nice day” has different sentiment with “what a nice day”. However, there are better calculations of feature values from word counts. The idea is that, if a word appear 10 times in a document, it is not 10 times more indicative than it appearing 1 time. You can choose to use tf/idf or BM25 method to calculate feature values.

The scikit-learn package has functions that directly convert a document to a feature vector. Please check sklearn.feature_extraction.text.

### 2.2 Word embedding

Word embedding is one method of encoding words into vectors. Embedding models often learn word embedding vectors from large corpora. These vectors often brings semantics in their relations. For example, $v_{king} - v_{male} + v_{female} \approx v_{queen}$. Here each $v$ is the vector representation of the work in subscript. The linear calculation here somehow represents the semantic relation of these words. Word embeddings often are trained first and then used in down-stream tasks, for example, our classification task.

In this project, the second method of converting a document into a vectors is to sum up word vectors in the sentence. Let a sentence be a list of words, $[j_1, j_2, \ldots, j_K]$. Each word $j_k$ has an embedding vector $v_{jk}$. Then the vector representation of the document is $\sum_{k=1}^{K} v_{jk}$. This method is called Sum of Word Embeddings (SOWE). See this this paper.

In this project, we will provide you GloVe word embeddings with 50 dimensions. You can choose to use other embeddings if you would like to explore a little more in this part.

### 3 Model training and testing

You should run model selection procedure to find the best hyper-parameter and use it in the task.

The procedure of model selection is to choose a learning algorithm and its hyper-parameter by checking the performance of the learned classifier on a single validation set or through the cross validation procedure.

**Requirements:** In this project, you are asked to try at least three different learning algorithms. For each algorithm, you need to tune its hyper-parameter(s), that is, you need to choose the hyper-parameter from a set of candidate values based the corresponding cross-validation performances of the classifiers trained with these candidate values. The candidate set for each learning algorithm should contain at least four settings.

Here is a guideline of how to decide the candidate set of hyper-parameters. Take logistic regression for example: its hyper-parameter $\lambda$ can take any positive values. Often the time, you want to test a set of hyper-parameters $\lambda_1 < \lambda_2 < \ldots < \lambda_K$ such that the model overfit the data
with the smallest value $\lambda_1$ but underfit the data with the largest value $\lambda_K$. In this way, you can get a sense that the best value is within the range your candidate values, and your final choice of the hyper-parameter will not deviated the best value radically.

`scikit-learn` provide functions for you to do cross-validation and hyper-parameter tuning. Please check `sklearn.model_selection.GridSearchCV`.

### 4 Submission requirements

You need to follow submission requirements in this section to guarantee that you submission can be evaluated smoothly.

In your submission, you need to include 1) a zip file containing your code, 2) your predictions, 3) your nickname in a file, and 4) and your report.

Your submitted code should clearly show your model selection procedure for each learning algorithm. In the documentation of your code, you should make clear how your code runs the model selection procedure for a specific learning algorithm and a possible set of candidate hyper-parameter values.

This time we require you to submit your predictions for the test set we have released. The file should have one prediction (0 or 1) for a review in a line. The snippet shows what the file should be like. The file should be the output of your classifier – you cannot make any manual changes.

```plaintext
0
1
0
1
...
```

As we will generate a leaderboard for the competition, we need your nicknames to show the performance of your classifiers. You may want a cool nickname but others cannot identify you from it.

Your report should clearly states what you have done for the project.

- Describe the two methods you have used to prepare the data, i.e. convert documents to vectors.

- State the learning algorithms you have used and the meaning of their hyperparameters. Note that, the hyperparameter settings in Python or matlab functions might be different from what we have talked in our classes. Taking logistic regression for example, the setting of $C$ parameter has equivalent effect of a special setting of our $\lambda$ parameter, but $C$ tunes the model in an opposite way as $\lambda$. You need to show the candidate set of hyper-parameters for each learning algorithm and the reason of your choices.

- Describe your model selection procedure. Your report should be clear enough for one to repeat your experiment by only reading your report.

- Plot training errors and validation errors against different settings of hyperparameters. For example, plot the two types of errors versus the value of $\lambda$ or $C$ in SVM, or the two types of errors versus number of iterations in AdaBoost. You should get curves like the ones in problem 4 of midterm exam. Do this for each learning algorithm and each classification task.
• Show the hyper-parameter setting for the final classifier you submit, and estimate a 95% confidence level of the performance of your final classifier. State how confident you are about the performance of your classifier being better the threshold defined below.

You are encouraged to describe any interesting findings from this project. Try to make the report pleasant to read – not only dry results.

5 Evaluation

In this project, your submission is evaluated based on your code, the final performance of your classifiers, and your report.

• You get 30% of full points if you have correctly used the two methods to prepare training and test data.

• You get 25% of full points if you have correctly done model selection procedure for hyper-parameter tuning for each classification task.

• You get 35% of full points if you have tried three different learning algorithms for each classification task.

• You get 20% of full points if your report clearly shows what you have done and satisfies the requirements stated in the last section.

If you do not get full points for any grading item, you get partial points according to our estimation of the amount of effort.