Coursework **Spring 2019**
Statistical Pattern Recognition (COMP136) **Roni Khardon**
Department of Computer Science, Tufts University **Rishit Sheth**

Programming Project 1

This assignment is due by Thursday, February 7, 7:30pm. **Please submit a hardcopy of the report (in class) and the code electronically (via provide).**

Overview: Unigram Model

Recall the unigram model from the first written assignment. A unigram model over a vocabulary of K words is specified by a discrete distribution with parameter $\boldsymbol{\mu}$. Under this model, the probability of the k-th word of the vocabulary appearing is given by μ_k. In this experiment, we will use the unigram model to evaluate different learning methods, and to perform model selection.

Task 1: Model Training, Prediction, & Evaluation

In class we developed three methods for prediction: using the maximum likelihood (ML) estimate, using the maximum a posteriori (MAP) estimate, and using the predictive distribution. In this part, we evaluate the effectiveness of the three different methods. More specifically, we will use text training data to perform unigram model learning according to each method and then calculate the *perplexity* of the learned models on a held-out test data set. Perplexity is a standard effectiveness metric in probabilistic language modeling for scoring how well a model predicts a given collection of words (low perplexity values imply good performance). Perplexity is defined by

$$PP = p(y_1, y_2, \cdots, y_N|\text{model})^{-\frac{1}{N}} \overset{\text{unigram}}{=} \exp\left(-\frac{1}{N}\sum_{i=1}^{N}\ln p(y_i)\right)$$

On the course webpage, you will find two files `training_data.txt` and `test_data.txt` each of size $N = 640,000$ words. We have pre-processed and "cleaned" the text so it does not require *any* further manipulation and you only have to read the space-separated strings from the corresponding ASCII text files.

For each size of training set in $\left[\frac{N}{128}, \frac{N}{64}, \frac{N}{16}, \frac{N}{4}, N\right]$, you should train a unigram model according to the three different methods discussed above (use the initial segment of the full training data). Use a Dirichlet distribution with parameter $\boldsymbol{\alpha} = \alpha'\mathbf{1}$ as a prior (this is a scalar α' multiplied by a vector of ones). Set $\alpha' = 2$ for this part. Prediction equations for these models are given below.

To avoid having words in the test set that are not in your vocabulary, start by building a vocabulary from the entire train and test sets and then use this vocabulary in Tasks 1 and 2 (normally, you would determine the size of the vocabulary from just the training set). You should find $K = 10,000$ distinct words.

When run, your code should report the perplexities on the train and test set under the three trained models for each train set size. Plot the results as a function of train set size (it is useful to plot all three methods together) and provide some observations. In your discussion of the results, please address the following:

- What happens to the test set perplexities of the different methods with respect to each other as the training set size increases? Please explain why this occurs.
- What is the obvious shortcoming of the maximum likelihood estimate for a unigram model? How do the other two approaches address this issue?
- For the full training set, how sensitive do you think the test set perplexity will be to small changes in α'? Why?

Task 2: Model Selection

Here we use the same data and vocabulary as in Task 1. In the previous part, we set the value of the hyperparameter, α', manually. In this part, you will use the evidence function (last question of the first written assignment) and training data to select a value of α'. In general, direct maximization of the evidence function to determine the hyperparameter can be difficult. Here we only have one hyperparameter and can therefore use a "brute-force" grid search to select α'.

In particular, compute the log evidence at $\alpha' = 1.0, 2.0, \ldots, 10.0$ for a training set of size $\frac{N}{128}$. Also, compute the perplexities on the test set (use the predictive distribution) at these same values. When run, your code should output the list of evidence and perplexity values for each α.

Plot the log evidence and test set perplexity as a function of α' and discuss what you see. In your discussion of the results, please address the following:

- Is maximizing the evidence function a good method for model selection on this dataset? Why?

Task 3: Author Identification

Can the unigram model identify authors? In this part, we apply the model to this problem. On the course web page you will find three additional files for this assignment. Each of them is a cleaned text version of a classic novel (thanks to the Gutenberg project).

To avoid having words in the test set that are not in your vocabulary, start by building a vocabulary from all three files and use this vocabulary in the experiments. You should find $K = 16,411$ distinct words.

Train the model on `pg345.txt.clean` (use the predictive distribution with $\alpha' = 2$) and evaluate the perplexity on each of the other two texts. When run, your code should output the resulting perplexities.

Please report the results. One of the test files is by the same author as the training file but the other is not. Was the model successful in this classification task?

Additional Notes

- Aside from standard I/O, math, and plotting libraries, **NO** external libraries should be used for this assignment.
- In Task 1, you'll need to handle $\ln(0) \triangleq -\infty$ as a special case in your code.
- Useful Equations:
  - Prediction using ML estimate: $p(y_* = w_k|\hat{\boldsymbol{\mu}}_{\text{ML}}) = \frac{m_k}{N}$
  - Prediction using MAP estimate: $p(y_* = w_k|\hat{\boldsymbol{\mu}}_{\text{MAP}}) = \frac{m_k+\alpha_k-1}{N+\alpha_0-K}$
  - Prediction using predictive distribution: $p(y_* = w_k|Y) = \frac{m_k+\alpha_k}{N+\alpha_0}$
  - Evidence: $\Pr(Y|\boldsymbol{\alpha}) = \frac{\Gamma(\alpha_0)\prod_{k=1}^{K}\Gamma(\alpha_k+m_k)}{\Gamma(\alpha_0+N)\prod_{k=1}^{K}\Gamma(\alpha_k)}$
  - In above, m_k = No. of times k-th word of vocabulary appears, N = Total no. of words, and $\alpha_0 = \sum_{k=1}^{K}\alpha_k$
  - $\Gamma(x) = (x-1)!$

Submission

- The implementation must be written in Matlab or Python.
- All your source code for the assignment should be submitted. Please write clear code with documentation as needed. The various parts of the implementation such as training, prediction, testing, and model selection should be *easily* identifiable. The source code should
  1. Run on *homework.eecs.tufts.edu*.
  2. Run without editing.
  3. Run with a single command (if there is more than one execution command required, include those commands in a single Bash script).
  4. Output the requested results.

  You can assume the data files will be available in the same directory as where the code is executed. Please include a short README file with the code execution command.
- For electronic submission, put all the files into a zip or tar archive, for example `myfile.zip` (you do not need to submit the data we give you). Please do not use another compression format such as RAR. Then submit using `provide comp136 pp1 myfile.zip`.
- Your assignment will be graded based on the clarity and correctness of the code, and presentation and discussion of the results.