Coursework **Fall 2018**
Machine Learning (B555) **Roni Khardon**
Department of Computer Science, IU Bloomington

Programming Project 1

This assignment is due by Wednesday 9/12, 11:59pm via Canvas.

You can write your code in any programming language so long as we are able to test it on SICE servers. While we will not test all submissions we will select a subset of assignments to test.

Data

Data for this assignment is provided in a zip file `pp1data.zip` on Canvas.

Overview: Unigram Model

Recall the unigram model from question 2 of written assignment 1. 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: (1) using the maximum likelihood estimate, (2) using the MAP estimate, and (3) using the predictive distribution. In this part, we evaluate the effectiveness of these 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

$$PP = p(w_1, w_2, \cdots, w_N|\text{model})^{-\frac{1}{N}} \overset{\text{unigram}}{=} \exp\left(-\frac{1}{N}\sum_{i=1}^{N}\ln p(w_i)\right)$$

In this part we use the data 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 (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) and 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 dictionary from the entire train and test sets and use this vocabulary in the experiments. You should find K=10000 distinct words.

When run, your code should report the perplexities on the train set (i.e., the current portion being trained on) 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 dictionary K=10000 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 (from written assignment 1) 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 α' and can therefore use a "brute-force" grid search to select its value.

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 log-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?

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 3 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 dictionary from all 3 files and use this vocabulary in the experiments. You should find K=18,251 distinct words.

Train the model on `pg121.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 perplexities

for the two other texts. 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? Please report and discuss these results.

Additional Notes

- You may use standard I/O, math, and plotting libraries. Other than these, please write all the code yourself without referring to special libraries or modules.
- In Task 1, you will need to handle $\ln(0) \triangleq -\infty$ as a special case in your code.
- Useful Equations:
  - Prediction using ML estimate: $p(\text{next word} = k\text{-th word of vocabulary}) = \frac{m_k}{N}$
  - Prediction using MAP estimate: $p(\text{next word} = k\text{-th word of vocabulary}) = \frac{m_k+\alpha_k-1}{N+\alpha_0-K}$
  - Prediction using predictive distribution: $p(\text{next word} = k\text{-th word of vocabulary}) = \frac{m_k+\alpha_k}{N+\alpha_0}$
  - Evidence: $\Pr(\text{Data}|\boldsymbol{\alpha}) = \frac{\Gamma(\alpha_0)\prod_{k=1}^{K}\Gamma(\alpha_k+m_k)}{\Gamma(\alpha_0+N)\prod_{k=1}^{K}\Gamma(\alpha_k)}$
  - In the above, m_k = Number of times the k-th word of the vocabulary appears in the document, N = Total number of words in the document, and $\alpha_0 = \sum_{k=1}^{K}\alpha_k$.
- Recall that $\Gamma(x) = (x-1)!$. Since our x values are integers you can calculate $\Gamma()$ values using factorial. To avoid numerical overflow, instead of calculating the factorial and then taking log, calculate the log directly.

Submission

Please write clear code with sufficient documentation so that we can read it. In addition write a README file that explains how the code is organized (if in multiple files) and how to compile and run the code. If this is non-trivial please write a script that runs the code and explain how to use it in the README file. When run in this manner your code should produce all the results and plots as requested above.

Please submit two items via Canvas: (1) Please write a report on the experiments, their results, and your conclusions as requested above. Prepare a PDF file with this report. (2) Collect all your code for the assignment (including the README file) in a zip file named `pp1code.zip`. You do not need to include the data that we provided. Your code should assume that the data files will have names as specified above and will reside in sub-directory pp1data/ of the directory where the code is executed.

Grading

Your assignment will be graded based on (1) the clarity of the code, (2) its correctness, (3) the presentation and discussion of the results.