CSCI 682 - Topics in Artificial Intelligence: Natural Language Processing
Assignment 2

Solutions to the questions on this assignment should be submitted via PDF and .py file to Canvas before **February 22nd at 11:59 pm**. Make sure to justify your answers.

I highly encourage you to collaborate with one another. However, you must write up your own solutions **independently**. Feel free to communicate via Discord and to post questions on the appropriate forum in Canvas. Do not post solutions. Also, please include a list of the people you work with at the top of your submission.

Have fun!

Problems

1. In this problem, we will implement logistic regression for binary classification step by step. The file `logRegSkeleton.py` on Canvas may be useful as a foundation.

   (a) (5 pts) Implement the `sigmoid(x)` function.

   (b) (8 pts) Implement the `predict_proba(X, w, b)` function which returns an estimate of $P(y = 1|x)$ for each observation in X.

   (c) (2 pts) Implement the `predict(X, w, b, thresh)` function which returns 0 or 1 for each observation in X.

   (d) (5 pts) Implement the `loss(X, y, w, b)` function which returns the average binary cross-entropy loss given true labels y.

   (e) (5 pts) Implement the `gradients(X, y, w, b)` function which returns the gradient of the loss with respect to the model weights and the bias term.

   (f) (5 pts) Implement the `fit(X, y)` function using gradient descent.

   (g) (5 pts) Generate a synthetic dataset by sampling model parameter values using the provided `generateData(n, d)` function. Think about appropriate values for n and d to test your logistic regression implementation and consider splitting the data into separate train and test sets. Train logistic regression models using at least three different values for η and plot the training loss as a function of epoch. Include a horizontal line on these figures showing the loss of the final trained model on the test set. Explain your choices and describe your observations.

(h) (5 pts) Extra Credit - Extend your implementation to handle multiclass classification problems.

2. (a) (5 pts) Write a function `perplexity(model, test)` that computes the perplexity of a language model on a test corpus. `model` should be a function that takes a context and a next word as input and returns the conditional probability of that word given the context under the model. `test` is a sequence of words over which perplexity should be computed.

   (b) (5 pts) Design a small training corpus ($\sim 5-10$ word instances) and a small test set ($\sim 3-5$ word instance). Make sure that all observations in the test set have non-zero probability under the models. Compute perplexity by hand for a unigram language model and a bigram language model given this data.

3. In this question, we will explore the performance of language models trained and tested in different ways. We will focus on the `t8.shakespeare.txt` and `alice29.txt` corpora available on Canvas.

   (a) (5 pts) To start, let `t8.shakespeare.txt` be our training data and `alice29.txt` be our test data. Use NLTK or spaCy to split each corpus into sentences and words. Use the training data to build a vocabulary and be careful to map unseen test words to a special `<UNK>` token.

   (b) (8 pts) Build unigram and bigram models using add-1 smoothing.

   (c) (7 pts) Use the scikit-learn implementation of multinomial logistic regression to train a bigram language model. Specifically, given a word w_{t-1}, this model should produce a probability distribution over next words w_t. Consider using HashingVectorizer with `analyzer=lambda x: x` and tokens from part (a) to generate word representations. Briefly explain why using an explicit one-hot encoding may be problematic in this case.

   (d) (5 pts) Evaluate the performance of these models on the test set using perplexity. Discuss your results.

   (e) (10 pts) Repeat the above process using (1) `alice29.txt` as the training data and `t8.shakespeare.txt` for testing, (2) `t8.shakespeare.txt` for both training and testing (be sure to perform an appropriate split), and (3) `alice29.txt` for both training and testing (be sure to perform an appropriate split).

   (f) (10 pts) Propose a different approach for splitting the corpora into train and test sets. Repeat $(a-d)$ above with this approach and compare the results.