Assignment- 0001 Probabilistic and Nearest Neighbor Methods in Textual Analysis

Dr. Mali

Due Date: 02/18/2024

1 Objective

The primary objective of this assignment is to provide an in-depth exploration and practical application of advanced statistical and machine learning methods in the field of Natural Language Processing (NLP). Students are expected to develop, implement, and critically analyze trigram-based probabilistic models, Naive Bayes classifiers, and K-Nearest Neighbors (KNN) classifiers, using the Brown Corpus as the primary dataset. The assignment aims to cultivate a robust understanding of the mathematical foundations of these methods and their practical implications in text analysis and classification.

1.1 Key Learning Outcomes

1. **Understanding of Trigram-Based Probabilistic Models:** Grasp the concept of N-grams in language modeling and their significance in predicting the likelihood of sequential elements in a text. Mathematically formulate trigram probabilities and comprehend the complexities involved in calculating and interpreting these probabilities. Address and overcome challenges like data sparsity and zero-probability using smoothing techniques.

2. **Proficiency in Naive Bayes Classification:** Master the principles of Bayesian probability and how the Naive Bayes classifier applies these principles for text categorization. Delve into the mathematical derivation of the Naive Bayes algorithm, understanding the role of prior and posterior probabilities in classification tasks. Analyze the implications of the independence assumptions of the Naive Bayes model.

3. **In-depth Knowledge of K-Nearest Neighbors (KNN) Methodology:** Understand the theory behind instance-based learning and the application of the KNN algorithm in text classification. Explore different distance metrics and their impacts on the classifier's performance. Evaluate the influence of the hyperparameter 'k' on model outcomes.

4. **Application of Statistical Inference and Maximum Likelihood Estimation:** Employ statistical inference techniques to draw conclusions from data, using MLE for optimizing model parameters. Integrate Bayesian inference for continuous model refinement.

5. **Comparative Analysis and Real-world Application:** Conduct a comparative analysis of trigram models, Naive Bayes, and KNN classifiers. Explore the potential of hybrid models combining these techniques. Develop critical thinking and analytical skills through the evaluation of models using metrics like accuracy, precision, recall, F1-score, and perplexity.

1.2 Assignment Structure

The assignment will be divided into theoretical exploration, practical implementation, and analytical comparison. Students will first develop a theoretical understanding of each method, followed by hands-on implementation using the Brown Corpus. The final phase involves a critical analysis of the models, including a comparative study and exploration of hybrid approaches.

1.3 Expected Outcome

Upon completion, students will have an advanced understanding of key NLP techniques, backed by strong mathematical foundations and practical experience in model development and evaluation. Slides and materials covered during class will be helpful while working on this assignment.

2 Part 1: Trigram Probabilistic Model - 5 points

Now, for this part, you can use NLTK to download data and preprocess data, but no other libraries besides numpy should be used in this phase.

2.1 Data Preprocessing and Trigram Formation

Begin with tokenization of the Brown Corpus and form trigrams, each consisting of three sequential tokens (x, y, z). Perform the following steps to download 10k sentences from the brown corpus: *The brown corpus is a foundational text resource in computational linguistics, consisting of a million words of American English texts from a wide range of sources. It's already categorized by genre, which can be helpful for more specific studies*

1. install NLTK in Python using pip or conda.
2. Use NLTK to download the Brown Corpus and extract sentences from it.
3. Each group is assigned a Team Alphabet, Team A = 1, Team B = 2, and so on. Now, you will extract 10k sentences from the brown corpus based on the following formulation.

Important: The assignment is not a group project; even though the dataset will be similar for students within a group but the data in validation and test sets will be different. Since you will perform random sampling without replacement to divide your corpus.

Now let's see mathematically how you will extract the data.

Given an array A and a desired number of sentences n to extract, we define a variable (x = Team number) that determines the extraction step. For a given x, the start and end indices for the extraction are calculated as follows:

- Start Index (S): $S = (x - 1) \times n$
- End Index (E): $E = x \times n - 1$

Thus, for a given x, the sentences are extracted from $A[S]$ to $A[E]$, inclusive. This can be represented as:

$$A_i = \text{extracted sentences from array } A, \text{ where } i = S, S + 1, \ldots, E$$

Where:

$$S = (x - 1) \times n$$
$$E = x \times n - 1$$

For example:

- When $Team = x = 1$, extract sentences from $A[0]$ to $A[9999]$.
- When $Team = x = 2$, extract sentences from $A[10000]$ to $A[19999]$, and so on.

2.2 Conditional Probability Estimation

Now, you will write a function in simple Python using only numpy to estimate the conditional probabilities.

Calculate the conditional probability $P(z|x, y)$ with frequency data, incorporating smoothing:

$$P_{\text{smooth}}(z|x, y) = \frac{\text{Count}(x, y, z) + \alpha}{\text{Count}(x, y) + \alpha \cdot N}$$

where N is the number of unique tokens.

2.3 Trigram Model

A trigram model considers a sequence of three words. Given a sequence of words $w_1, w_2, \ldots, w_n$, the probability of word w_n occurring after words w_{n-2} and w_{n-1} is modeled as:

$$P(w_n|w_{n-2}, w_{n-1})$$

The joint probability of a word sequence in the trigram model is given by:

$$P(w_1, w_2, \ldots, w_n) = \prod_{i=3}^{n} P(w_i|w_{i-2}, w_{i-1})$$

2.4 Estimation of Trigram Probabilities

The probabilities are typically estimated from a corpus of text. For a trigram (w_{i-2}, w_{i-1}, w_i), its probability is estimated as:

$$P(w_i|w_{i-2}, w_{i-1}) = \frac{\text{Count}(w_{i-2}, w_{i-1}, w_i)}{\text{Count}(w_{i-2}, w_{i-1})}$$

where $\text{Count}(w_{i-2}, w_{i-1}, w_i)$ is the number of times the trigram occurs in the corpus, and $\text{Count}(w_{i-2}, w_{i-1})$ is the number of times the preceding bigram occurs.

2.5 Smoothing Techniques

In practice, many trigrams will not occur in the corpus, leading to zero probability estimates. Smoothing techniques, such as Laplace smoothing, are employed to handle such cases:

$$P_{\text{Laplace}}(w_i|w_{i-2}, w_{i-1}) = \frac{\text{Count}(w_{i-2}, w_{i-1}, w_i) + \alpha}{\text{Count}(w_{i-2}, w_{i-1}) + \alpha \times V}$$

where V is the vocabulary size and α is a small constant.

2.6 Application of Bayesian Inference and MLE

Now, you will write a function in simple Python using only numpy to estimate your likelihood and posterior probabilities using a model optimized on the objective function .

Implement Bayesian methods for refining probability estimates and utilize Maximum Likelihood Estimation (MLE) for model optimization:

$$\hat{\theta} = \arg\max_{\theta} L(\theta; \text{data})$$

where $L(\theta; \text{data})$ represents the likelihood of the data under parameters θ.

Now, let's understand why we will reach true distribution using or optimizing the model using MLE.

Things to do: We have provided 5 formulations for the Bayes Rule. Comment on the importance of each form and select one that will be used in your codebase. Provide detailed comments stating why specific formulations can and cannot be used to estimate MLE and optimize your model.

We will now briefly explain each concept.

2.6.1 Bayesian Inference

Bayesian inference is a method of statistical inference which uses Bayes' theorem to update the probability for a hypothesis as more evidence or information becomes available. Bayes' theorem is mathematically stated as:

$$P(H|D) = \frac{P(D|H) \cdot P(H)}{P(D)}$$

where:

- $P(H|D)$ is the posterior probability of hypothesis H given the data D.

- $P(D|H)$ is the likelihood of the data D under the hypothesis H.
- $P(H)$ is the prior probability of the hypothesis H.
- $P(D)$ is the probability of the data D (the evidence).

2.6.2 Odds Form of Bayes' Theorem

The odds form is useful in decision theory and expresses the theorem as:

$$\frac{P(H|E)}{P(\neg H|E)} = \frac{P(E|H)}{P(E|\neg H)} \times \frac{P(H)}{P(\neg H)}$$

where $P(\neg H)$ denotes the probability of the hypothesis being false.

2.6.3 Bayesian Updating (Sequential Form)

Bayesian updating modifies beliefs with new information, sequentially applying Bayes' theorem:

$$P(H|E_1, E_2) = \frac{P(E_2|H, E_1) \times P(H|E_1)}{P(E_2|E_1)}$$

2.6.4 Generalized Bayes' Theorem for Multiple Hypotheses

For multiple hypotheses, Bayes' theorem is generalized as:

$$P(H_i|E) = \frac{P(E|H_i) \times P(H_i)}{\sum_j P(E|H_j) \times P(H_j)}$$

This is particularly useful in model selection or classification problems.

2.6.5 Bayesian Factor

Bayes' theorem in terms of the Bayesian factor (BF) for hypothesis comparison:

$$\text{BF} = \frac{P(E|H_1)}{P(E|H_0)}$$

BF measures how much the evidence favors one hypothesis over another.

2.6.6 Continuous Form

For continuous distributions, the theorem is expressed as:

$$P(H|E) = \frac{p(E|H) \times P(H)}{\int p(E|H')dH'}$$

Here, $p(E|H)$ is the likelihood as a probability density function.

2.7 Maximum Likelihood Estimation (MLE)

This part will also be developed using numpy, including dividing your corpus in train, test and validation.

MLE is a method for estimating the parameters of a statistical model by maximizing the likelihood function. The likelihood function $L(\theta; x)$ is a function of the parameters θ given the data x. MLE finds the value of θ that maximizes this function. The likelihood is often represented as the product of the probabilities of the observed data:

$$L(\theta; x) = \prod_{i=1}^{n} P(x_i|\theta)$$

The MLE $\hat{\theta}$ is defined as:

$$\hat{\theta} = \arg\max_{\theta} L(\theta; x)$$

2.7.1 Lemma: Consistency of MLE

Lemma: The MLE is consistent. As the sample size n grows, the MLE $\hat{\theta}$ converges in probability to the true parameter θ_0.

$$\lim_{n\to\infty} P(|\hat{\theta}_n - \theta_0| < \epsilon) = 1 \text{ for every } \epsilon > 0$$

One can easily prove the above lemma to get a convergence guarantee of the model.

2.8 Model Evaluation through Perplexity

This part will be developed using numpy, including metrics such as Accuracy, precision, recall and F1 score.

Evaluate the model using Perplexity (PPL), defined as:

$$\text{PPL}(W) = P(w_1, w_2, \ldots, w_N)^{-\frac{1}{N}}$$

where W is the word sequence and N is its length.

Definition of Perplexity

Perplexity of a language model on a given text is defined as the inverse probability of the test set, normalized by the number of words. For a sequence of words $w_1, w_2, \ldots, w_N$, the perplexity is given by:

$$\text{PPL}(W) = P(w_1, w_2, \ldots, w_N)^{-\frac{1}{N}}$$

where N is the length of the sequence, and $P(w_1, w_2, \ldots, w_N)$ is the probability of the word sequence as assigned by the model.

Perplexity of Trigram Models

For trigram models, perplexity can be further expanded as:

$$\text{PPL}(W) = \sqrt[N]{\prod_{i=3}^{N} \frac{1}{P(w_i|w_{i-2}, w_{i-1})}}$$

Here, $P(w_i|w_{i-2}, w_{i-1})$ is the conditional probability of word w_i given its preceding two words in the trigram model.

Interpreting Perplexity

Lower perplexity indicates a better model, implying that the model predicts the test data with higher accuracy. A perplexity of 1 would mean perfect prediction, though this is an ideal and not practically achievable.

Calculating Perplexity in Practice

In practice, due to the small magnitude of probabilities in text, it's common to compute perplexity in terms of log probabilities to avoid numerical underflow:

$$\text{PPL}(W) = \exp\left\{-\frac{1}{N}\sum_{i=3}^{N} \log P(w_i|w_{i-2}, w_{i-1})\right\}$$

3 Part 2: K-Nearest Neighbors (KNN) Classifier – 3 points

In this part, you will build an unsupervised learning algorithm known as K-Nearest Neighbours (KNN) using numpy to classify text extracted in part A. Now, we will briefly explain what KNN is and how it works.

3.1 Feature Engineering

Extract and normalize numerical features from the text data for the KNN classifier.

3.2 KNN Mathematical Framework

Implement the KNN classifier, determining the optimal number of neighbors (k) empirically. Classification is based on the majority vote of k nearest neighbors:

$$\text{Class}(x) = \text{mode}\{\text{Class}(x_1), \text{Class}(x_2), \ldots, \text{Class}(x_k)\}$$

3.2.1 Basic Principle of KNN

In KNN, an object is classified by a majority vote of its neighbors, with the object being assigned to the class most common among its k nearest neighbors. Mathematically, for a given data point x, the KNN classifier looks at the k closest points $x_1, x_2, \ldots, x_k$ and performs a majority vote to determine the class of x:

$$\text{Class}(x) = \text{mode}\{\text{Class}(x_1), \text{Class}(x_2), \ldots, \text{Class}(x_k)\}$$

3.2.2 Distance Metrics

The choice of the distance metric is crucial in KNN. Common distance metrics include:

- Euclidean Distance: $d(x, y) = \sqrt{\sum_{i=1}^{n}(x_i - y_i)^2}$
- Manhattan Distance: $d(x, y) = \sum_{i=1}^{n} |x_i - y_i|$
- Minkowski Distance: $d(x, y) = (\sum_{i=1}^{n} |x_i - y_i|^p)^{\frac{1}{p}}$

Provide a detailed analysis of which works in your scenario and why. Explain in terms of finding clusters, easier to find optimal parameters, how it works on outlier etc.

3.2.3 Choosing the Right k

Selecting the optimal value of k is critical. A smaller k makes the algorithm sensitive to noise, while a larger k makes it computationally expensive and may lead to misclassification due to the inclusion of points from other classes.

3.2.4 Weighted KNN

In weighted KNN, weights are assigned to the contributions of the neighbors, so that nearer neighbors contribute more to the classification than the more distant ones. A common weighting scheme is the inverse distance weighting:

$$w_i = \frac{1}{d(x, x_i)^2}$$

3.3 Classification Exercise

Now, define a categorization task using the Brown Corpus text extracted in Task A and apply the KNN classifier using Weighted KNN.

3.4 Classifier Performance Evaluation

Finally, you will evaluate the classifier using accuracy, precision, recall, and F1-score metrics.

4 Combined Analysis – 2 points

4.1 Integrated Mathematical Analysis

Investigate the synergy between the probabilistic model and the KNN classifier, discussing the theoretical implications of their combination. In simple words, explain the difference between the probabilistic model and the unsupervised model. Your answers should mathematically make sense.

5 Deliverables

- You will upload the source code (complete Python file) that performs preprocessing, estimates trigram probability, performs classification, and evaluation metrics.
- A comprehensive report detailing mathematical formulations, implementation, evaluation methodology, and insights or challenges.
- **Important: You can only use libraries such as numpy, ntlk and matplotlib, using sklearn/pytorch/jax/TensorFlow etc will lead to zero points.**
- We will provide partial grades in scenarios where the solution is incorrect.