Assignment A

CSE 517: Natural Language Processing - University of Washington

Winter 2023

Please consult the course website for current information on the due date, the late policy, and any data you need to download for this assignment. This assignment is designed to advance your understanding of text classification, feature design and selction, the evaluation of classifiers (e.g., the F_1 score), the mathematics of some important classification models, frequency information in text data, and some mathematical properties of language models.

Submit: You will submit your writeup (a pdf) and your code (do not include data) via Gradescope. Instructions can be found here. Note that you will make two submissions: one for the pdf, one for the code.

1 Text Classification – Eisenstein 4.6 (p. 89)

In the directory `review_polarity`, you will find a dataset of positively and negatively classified reviews that was used by Pang and Lee [2], a seminal paper about sentiment classification. Consult the readme file for more information. Hold out a randomly selected 400 reviews as a test set.

Sentiment lexicon-based classifier. Create a classifier using a sentiment lexicon. A lexicon from Hu and Liu [1] is provided in the directory `opinion_lexicon_English`, but you are welcome to find and use (with attribution, of course) another. Tokenize the data, and classify each document as positive if and only if it has more positive sentiment words than negative sentiment words. Compute and report the accuracy and F_1 score (on detecting positive reviews) on the test set, using this lexicon-based classifier.

Logistic regression classifier. Train a (binary) logistic regression classifier on your training set using features of your own choosing, and report its accuracy and F_1 score (as above) on the test set. In your write-up, describe the features you have chose and explain the reasoning behind your choice.

Do not use pretrained word vectors or any features implemented or constructed by anyone else. Do not use an existing implementation of logistic regression, stochastic gradient descent, or automatic differentiation.

Breaking good. For each of the following, write a review document that you believe would be considered as *positive* by human English speakers, and:

- your lexicon classifier predicts it as *positive*, whereas your logistic regression classifier predicts it as *negative*.
- your lexicon classifier predicts it as *negative*, whereas your logistic regression classifier predicts it as *positive*.
- both of your classifiers predict it as *negative*.

For each of the above scenarios, briefly discuss why your classifier(s) would make incorrect predictions for the document you created.

Statistical significance (extra credit). Determine whether the differences in accuracy and F_1 score are statistically significant at $\alpha = 0.05$, using two-tailed hypothesis tests: binomial for the difference in accuracy and bootstrap for the difference in macro F_1 score. Report the results.

Important note: You should implement all parts of this problem from scratch (you may use `numpy`). Do not use existing implementations for text tokenization, feature construction, logistic regression, stochastic gradient descent, automatic differentiation, or statistical significance testing. In general, it's a good idea to use existing, trusted implementations, but in this assignment we want you to experience attempting them on your own, even if your implementation is not the best in the world, so that you will fully grasp the nuts and bolts of these important ideas. If you aren't sure about whether it's okay to import a particular library, please **ask** on the discussion board!

2 Regularization – Eisenstein 2.5 (p. 44)

Suppose you are given two labeled datasets D_1 and D_2, with the same features and labels.

- Let $\boldsymbol{\theta}^{(1)}$ be the unregularized logistic regression (LR) coefficients from training on dataset D_1.
- Let $\boldsymbol{\theta}^{(2)}$ be the unregularized LR coefficients (same model) from training on dataset D_2.
- Let $\boldsymbol{\theta}^*$ be the unregularized LR coefficients from training on the combined dataset $D_1 \cup D_2$.

Under these conditions, prove that for any feature j,

$$\theta_j^* \geq \min\left(\theta_j^{(1)}, \theta_j^{(2)}\right)$$
$$\theta_j^* \leq \max\left(\theta_j^{(1)}, \theta_j^{(2)}\right).$$

3 XOR – Eisenstein 3.4 (p. 65)

Design a feedforward network to compute this function, which is closely related to XOR:

$$f(x_1, x_2) = \begin{cases} -1 & \text{if} \quad x_1 = 1 \wedge x_2 = 1 \\ 1 & \text{if} \quad x_1 = 1 \wedge x_2 = 0 \\ 1 & \text{if} \quad x_1 = 0 \wedge x_2 = 1 \\ -1 & \text{if} \quad x_1 = 0 \wedge x_2 = 0 \end{cases}$$

Your network should have a single output node that uses the "sign" activation function,

$$\text{sign}(x) = \begin{cases} 1 & \text{if} \quad x > 0 \\ -1 & \text{if} \quad x \leq 0 \end{cases}$$

Use a single hidden layer, with ReLU activation functions. Describe all weights and offsets.

4 Extra Credit: Initialization at Zero – Eisenstein 3.5 (p. 65)

Consider the same network as in problem 3 (with ReLU activations for the hidden layer), with an arbitrary differentiable loss function $\ell(y^{(i)}, \tilde{y})$, where $\tilde{y}$ is the activation of the output node. Suppose all weights and offsets are initialized to zero. Show that gradient descent will not learn the desired function from this initialization.

5 Substitution Cipher Breaking

This problem deals with substitution ciphers. A character-level substitution cipher corresponds to a map *encrypt* : $\Sigma \rightarrow \Sigma$, where Σ is the set of characters you use to write your texts. [1] Here is an example:

characters:	`abcdefghijklmnopqrstuvwxyz.` (space)
encrypt$(\cdot)$:	`hiwxyzpqjklmnovabcdefgrstu.` (space)

Note that the period and space symbols map to themselves. To encrypt a string in Σ^*, we encrypt each symbol in turn and concatenate. So, using this substitution cipher, we would map "`the zoo seems to be closed today.`" to "`eqy uvv dyynd ev iy wmvdyx evxht.`"

In the following questions, let $N = |\Sigma|$.

1. How many substitution ciphers are there?

2. In a given cipher, a symbol $x \in \Sigma$ is said to be *fixed* if *encrypt*$(x) = x$. (In our example above, the period and space symbols are fixed.) How many substitution ciphers are there with no fixed symbols? (Hint: the combinatorial concept of a *derangement* is useful to use here; we suggest you look it up. Make sure to show all steps in your work.)

3. Suppose you select a substitution cipher with *no fixed symbols*, uniformly at random, and your friend decides to try to break the cipher by selecting ciphers at random. What is the probability that your friend guesses your cipher *exactly* within k tries (a) if they assume nothing, and (b) if they (correctly) assume your cipher has no fixed symbols?[2]

4. A well-known problem with substitution ciphers is that, in a long ciphertext, the frequency of *encrypt*(x) will be close to the frequency of x in plaintext. In natural languages, there is a lot of variance in different symbols' frequencies. (For words, "Zipf's Law" states that the probability of the rth most frequent word in a text corpus will have relative frequency proportional to $\frac{1}{r}$.) Your task is to exploit this problem to decrypt the ciphertext we provide to you (`AA.encrypted.txt`). The original plaintext is in all-upper-cased English. You may assume the space symbol, numerical digits, and any other non-alphabetic symbols are fixed. This implies that $N = 26$. We suggest that you automate the calculation and visualization of the symbol frequencies. Some manual search may be required since your ciphertext (and any plaintext you use to estimate English relative frequencies) will be finite, leading to variance in your estimates. You should submit your decrypted text.

5. One way to make a cipher more robust to the analysis discussed above is to augment the output alphabet with more symbols: *encrypt* : $\Sigma \rightarrow \Sigma \cup \Gamma$. Then, during encryption, instances of more frequent input symbols can be randomly assigned to multiple output symbols. For example, suppose that `e` is the most frequent symbol in a plaintext corpus, accounting for 5% of tokens. We would then allocate $0.05|\Sigma \cup \Gamma|$ symbols to `e`, any of which might be chosen uniformly at random when encoding

[1] For questions 1–4, assume the mapping is one-to-one.

[2] This problem was inspired by the Enigma machine.

e. If we do this for every symbol in Σ, then the frequencies of all symbols in $\Sigma \cup \Gamma$ will be roughly equal in a ciphertext. If we require that the encoding function be *deterministic*, how would you design this new cipher to prevent frequency analysis attacks?

6 Valid Probabilities (n-Gram) – Based on Eisenstein 6.1 (p. 135)

Prove that n-gram language models give valid probabilities if the n-gram probabilities are valid. Specifically, assume that

$$\sum_{v \in \mathcal{V}} p(X_t = v \mid X_{t-n+1} = x_{t-n+1}, \ldots, X_{t-2} = x_{t-2}, X_{t-1} = x_{t-1}) = 1 \tag{1}$$

for all contexts $\langle x_{t-n+1}, \ldots, x_{t-2}, x_{t-1} \rangle$. For $m \geq 1$, let $\mathcal{V}^m \subset (\mathcal{V} \setminus \{\text{⯃}\})^*$ denote the set of sequences of length m that include no stop symbols. Prove that $\sum_{\boldsymbol{x} \in \mathcal{V}^m} p_{\mathit{model}}(\boldsymbol{x}) = 1$, where $m \geq 1$ and $p_{\mathit{model}}(\boldsymbol{x})$ is the probability of $\boldsymbol{x}$ under the n-gram model. Your proof should proceed by induction. You should handle the start-of-string case where the context is $n-1$ start symbols, but do not include the stop symbol.

Extra Credit: If we do not include the stop symbol, n-gram models define a valid probability distribution over sequences of a fixed length, as shown above. As a consequence, they do not define a valid probability distribution over $\mathcal{V}^*$. Show that by including the stop symbol, n-gram models define a valid distribution over sequences of all lengths. That is, show $\sum_{\boldsymbol{x} \in \mathcal{V}^\dagger} p_{\mathit{model}}(\boldsymbol{x}) = 1$.

7 Valid Probabilities (RNN) – Based on Eisenstein 6.2 (p. 135)

First, show that RNN language models are valid using a similar proof technique to the one in problem 6.

Next, let $p_r(\boldsymbol{x})$ indicate the probability of $\boldsymbol{x} \in \mathcal{V}^m$ under RNN r. An ensemble of R RNN language models computes the probability:

$$p(\boldsymbol{x}) = \frac{1}{R} \sum_{r=1}^{R} p_r(\boldsymbol{x}) \tag{2}$$

Does an ensemble of RNN language models compute a valid probability? Prove your answer.

References

[1] Minqing Hu and Bing Liu. Mining and summarizing customer reviews. In *Proc. of KDD*, 2004.

[2] Bo Pang and Lillian Lee. A sentimental education: Sentiment analysis using subjectivity summarization based on minimum cuts. In *Proc. of ACL*, 2004.