COMS 4721/4771 HW1 (Spring 2024)

Due: Sun Feb 11, 2024 at 11:59pm

This homework is to be done **alone**. No late homeworks are allowed. To receive credit, a typesetted copy of the homework pdf must be uploaded to Gradescope by the due date. You must show your work to receive full credit. Discussing possible approaches for solutions for homework questions is encouraged on the course discussion board and with your peers, but you must write your own individual solutions and **not** share your written work/code. You **must cite** all resources (including online material, books, articles, ai generative bots, help taken from/given to specific individuals, etc.) you used to complete your work.

It is your responsibility to protect your work, and it is a violation of academic integrity policy to post any part of these questions or your answers to public websites such as: github, bitbucket, chegg, coursehero, etc. **Violators will be reported to the dean for disciplinary action**.

1 Maximum Likelihood Estimation

(a) Consider the density $p(x \mid \theta) := \begin{cases} e^{\theta - x} & \text{if } x \geq \theta \\ 0 & \text{otherwise} \end{cases}$ for some $\theta > 0$. Suppose that n samples $x_1, x_2, \ldots, x_n > 0$ are drawn i.i.d. from $p(x \mid \theta)$. What is the MLE of θ given the samples?

Hint: In class we computed the MLE by finding the log-likelihood, taking the gradient (with respect to θ), and setting it equal to 0. However, this only works well for differentiable concave log-likelihood functions. You should never forget that the end goal is simply to maximize the likelihood (or log-likelihood) function.

(b) Show that for the MLE θ_{MLE} of a parameter $\theta \in \mathbb{R}^d$ and any known bijective function g, the MLE of $g(\theta)$ is $g(\theta_{\text{MLE}})$. It turns out that this holds for any function g, i.e. g need not be bijective (you do not need to show this but you should try to understand why this is true). Use this fact to find the MLE of e^θ in the setting of part (a).

(c) Sometimes we have prior knowledge concerning the value of the parameter θ. This is often encoded as a prior distribution characterized by $p(\theta)$. In such cases one usually computes the MAP (maximum a posteriori) estimate of θ, defined as $\theta_{\text{MAP}} = \arg\max_\theta p(\theta \mid x_1, x_2, \ldots, x_n)$, instead of the MLE.

In the setting of part (a) suppose that we know from prior information that θ is likely small. Specifically, we model this with the prior $p(\theta) = 2e^{-\theta^2}\pi^{-1/2}$. Compute the MAP estimate of θ.

Hint: First show that $\theta_{\text{MAP}} = \arg\max_\theta p(x_1, x_2, \ldots, x_n \mid \theta)p(\theta)$.

2 Analyzing iterative optimization

Minimizing an objective function is of central importance in machine learning. In this problem, we will analyze the an iterative approach for finding $\beta \in \mathbb{R}^d$ that (approximately) minimizes $\|A\beta - b\|_2^2$ for a given matrix $A \in \mathbb{R}^{n \times d}$ and vector $b \in \mathbb{R}^n$.

Consider the following iteration approximation algorithm:

- Initially, $\beta^{(0)} = (0, \ldots, 0) \in \mathbb{R}^d$ is the zero vector in $\mathbb{R}^d$.
- For $k = 1, 2, \ldots, N$:
  - Compute $\beta^{(k)} := \beta^{(k-1)} + \eta A^\mathsf{T}(b - A\beta^{(k-1)})$.

In above, $\eta > 0$ is a fixed positive number usually referred to as the step size, and N is the total number of iterations. Define $M := A^\mathsf{T}A$ and $v := A^\mathsf{T}b$.

(i) Show that the matrix M is symmetric positive semi-definite.

Throughout, assume that the eigenvalues of M, denoted by $\lambda_1, \ldots, \lambda_d$, satisfy $\lambda_i < 1/\eta$ for all $i = 1, \ldots, d$.

(ii) Prove (e.g., using mathematical induction) that, for any positive integer N,

$$\beta^{(N)} = \eta \sum_{k=0}^{N-1} (I - \eta M)^k v.$$

(Here, for a square matrix B, we have $B^0 = I$, $B^1 = B$, $B^2 = BB$, $B^3 = BBB$, and so on.)

(iii) What are the eigenvalues of $\eta \sum_{k=0}^{N-1} (I - \eta M)^k$? Give your answer in terms of $\lambda_1, \ldots, \lambda_d$, η, and N.

(iv) Let $\hat{\beta}$ be any non-zero vector in the range of M satisfying $M\hat{\beta} = v$. Prove that

$$\|\beta^{(N)} - \hat{\beta}\|_2^2 \leq e^{-2\eta\lambda_{\min} N} \|\hat{\beta}\|_2^2,$$

where $\lambda_{\min}$ is the smallest non-zero eigenvalue of M.

Hint: You may use the fact that $1 + x \leq e^x$ for any $x \in \mathbb{R}$.

This implies that as the number of iterations N increases, the difference between our estimate $\beta^{(N)}$ and $\hat{\beta}$ decreases exponentially!

3 Bayes Error Rate

Consider the classification problem on an arbitrary (measurable) input space X and a binary output space $Y = \{0, 1\}$. Given a joint data distribution $\mathcal{D}$ over $X \times Y$, let $g : X \to Y$ denote the Bayes classifier $g(x) := \arg\max_Y \Pr[Y|X = x]$. Define $\text{ERR}(g) := \Pr_{(x,y)\sim\mathcal{D}}[g(x) \neq y]$ as the error rate of the Bayes classifier. Prove the following statements about the error rate of g.

(i) Prove that

$$\text{ERR}(g) = \frac{1}{2} - \frac{1}{2}\,\mathbb{E}_{x\sim\mathcal{D}|_X} \Big|2\eta(x) - 1\Big|,$$

where $\mathcal{D}|_X$ denotes the marginal distribution on X, and $\eta(x) := P[Y = 1|X = x]$.

(ii) Let $p_1 := \Pr[Y = 1]$, then

$$\text{ERR}(g) = \int_{x\in X} \min\Big\{(1-p)f_0(x), pf_1(x)\Big\}dx,$$

where f_1 and f_0 are densities of the class conditional distributions $\Pr[X|Y = 1]$ and $\Pr[X|Y = 0]$ respectively.

(iii) If the class priors are equal (that is, $\Pr[Y = 0] = \Pr[Y = 1] = 1/2$), then

$$\text{ERR}(g) = \frac{1}{2} - \frac{1}{4} \int_{x\in X} \Big|f_1(x) - f_0(x)\Big|dx,$$

where f_1 and f_0 are densities of the class conditional distributions $\Pr[X|Y = 1]$ and $\Pr[X|Y = 0]$ respectively.

4 Exploring the Limits of Current Language Models

Recently, OpenAI launched their Chat Generative Pre-Trained Transformer (ChatGPT), a powerful language model trained on top of GPT-3. Models like ChatGPT and GPT-3 have demonstrated remarkable performance in conversation and Q&A and overall large generative modeling. Despite their impressive performance, the sentences produced by such models are not "foolproof". Here we will explore how effectively can one distinguish between human vs. AI generated written text. The classification model which you will develop can also be used to generate new sentences in a similar fashion to ChatGPT!

Suppose we have a training corpus $\mathcal{D}$ with M documents, each composed of at least one sentence and labelled with a class y. Let M_y denote the number of documents labelled y. One of the simplest language models for text classification is N-grams. An N-gram is a sequence of N consecutive words $(w_{1:N})$. The idea behind the model is that the probability of a word w_i appearing depends only the $N-1$ words before it. This local independence assumption can be considered as a Markov-type property. The following section walks through the derivation of the bigram ($N=2$) model classifier.

Let document $D=(w_{1:n})$ be a sequence of n words. Given a class y, the bigram local independence assumption states that the probability of D appearing is

$$P(w_{1:n} \mid y) = \prod_{i=1}^{n} P(w_i \mid w_{i-1}, y).$$

We use maximum likelihood estimation (MLE) to determine the conditional probabilities above. Let $C(w_{i-1}w_i \mid y)$ denote the number of times (counted with repeats) the bigram (w_{i-1}, w_i) appears amongst all documents with class label y. Then, we can approximate the conditional probability as

$$P(w_i \mid w_{i-1}, y) = \frac{P(w_{i-1}w_i \mid y)}{P(w_{i-1} \mid y)} \underset{\text{approx.}}{=} \frac{C(w_{i-1}w_i \mid y)}{C(w_{i-1} \mid y)}.$$

The probability that the document is labelled as class y is

$$P(y \mid w_{1:n}) = \frac{P(y, w_{1:n})}{P(w_{1:n})} = \frac{P(y)P(w_{1:n} \mid y)}{\sum_{\tilde{y}} P(\tilde{y})P(w_{1:n} \mid \tilde{y})},$$

where $P(y) = M_y/M$ is the prior. Formally, the bigram classifier is

$$f(w_{1:n}) = \arg\max_y P(y \mid w_{1:n}) = \arg\max_y P(y)P(w_{1:n} \mid y).$$

To address *out of vocabulary* (OOV) words (i.e. N-grams in the test set but not in the training corpus), N-gram models often employ "smoothing". One common technique is Laplacian smoothing, which assumes that every N-gram appears exactly one more time than actually observed. Given a vocabulary V, the conditional probability becomes

$$P(w_i \mid w_{i-1}) \underset{\text{approx.}}{=} \frac{C(w_{i-1}w_i) + 1}{C(w_{i-1}) + |V|}.$$

The vocabulary V is the set of all unique words which appear across all documents in the entire corpus — not just the training set. Note that the denominator is increased by $|V|$ so the probabilities sum to one.

(i) Calculate the class probability $P(y \mid w_{1:n})$ for a trigram ($N=3$) model with Laplacian smoothing.

In the following parts, you will create a bigram and trigram model to classify if documents are written by humans or by ChatGPT. Download the datafile `humvgpt.zip`[1]. This file contains around $40,000$ human written and $20,000$ ChatGPT written documents. The data is stored in separate text files named `hum.txt` and `gpt.txt` respectively.

(ii) (a) Clean the data by removing all punctuation except "`,.?!`" and converting all words to lower case. You may also find it helpful to add special `<START>` and `<END>` tokens to each document for calculating N-gram probabilities. Partition the initial 90% of the data to a training set and final 10% to test set (the data has already been shuffled for you).

[1] The data is adapted from https://huggingface.co/datasets/Hello-SimpleAI/HC3.

(b) Train a bigram and trigram model by calculating the N-gram frequencies per class in the training corpus. Calculate and report the percentage of bigrams/trigrams in the test set (counted with repeats) that do not appear in the training corpus (this is called the OOV rate). You should calculate two OOV rates, one for bigrams and one for trigrams.

(you must submit your code to get full credit)

(c) Evaluate the two models on the test set and report the classification accuracy. Which model performs better and why? Your justification should consider the bigram and trigram OOV rate.

This study will also tell you how difficult or easy it is to distinguish human vs. AI generated text!

Besides classification, N-gram models may also be used for text generation. Given a sequence of $n-1$ previous tokens $w_{i-n+2:i}$, the model selects the next word according to the probability distribution,

$$P(w_{i+1} = w) \underset{\text{approx.}}{=} \frac{\exp(C(w_{i-n+2:i}w)/T)}{\sum_{\tilde{w}} \exp(C(w_{i-n+2:i}\tilde{w})/T)}.$$

In the above equation, T is referred to as the temperature.

(iii) (a) Using $T = 50$, generate 5 sentences with 20 words each from the human and ChatGPT bigrams/trigrams (you should have 20 sentences total using $T = 50$). Which text corpus and N-gram model generates the best sentences? What happens when you increase or decrease the temperature? You should begin generation with $N-1$ `<START>` tokens and stop once you generate the `<END>` token.

(b) You should notice that your generated sentences do not preserve long-range context. Nowadays, language models (e.g. transformers) use an *attention mechanism* to capture this context. Why do sentences generated from the n-gram model often fail in this regard?