CS 533: Natural Language Processing **(Due: 02/04/20)**

Assignment 1

Instructor: Karl Stratos

- 3 problems: total 32 points (11 + 9 + 12)
- No collaboration
- Due by 11:59pm of the due date, no late submission accepted
- Use the provided LaTeX assignment template to write the answers. Upload the code as well.

Problem 1: Preliminaries

((1+1+1+1) + (1+1+1+1) + (1+1+1) = 11 points)

1. **(Probability)**: The **expected value** of a scalar discrete random variable $X \in \mathcal{X}$ ($\mathcal{X}$ is the set of all possible values that X can take) with a distribution p_X is defined as

$$\mathbf{E}[X] = \mathop{\mathbf{E}}_{x \sim p_X}[x] := \sum_{x \in \mathcal{X}} p_X(x)x$$

We will write $\mathbf{E}[X]$ when it's clear which distribution the expectation is with respect to, and add $x \sim p_X$ when we want to make it explicit. $\mathbf{E}[X]$ is also called the **mean** of p_X and denoted as $\mu_X \in \mathbb{R}$. The **variance** of p_X is defined as $\mathrm{var}(X) := \mathbf{E}\left[(X - \mu_X)^2\right]$. For this problem only, we will assume log base 2 (i.e., $\log 2 = 1$).[1] The **entropy** of p_X is defined as

$$H(p_X) := \mathop{\mathbf{E}}_{x \sim p_X}[-\log p_X(x)]$$

Note that if there is an element $x \in \mathcal{X}$ with probability zero, the log term in entropy diverges (i.e., $\log 0$ is undefined). An important convention in calculating entropy is to ignore such elements by treating $0 \log 0$ as zero. For instance, if $\mathcal{X} = \{1, 2, 3\}$ and $p_X(1) = p_X(3) = 0.5$ and $p_X(2) = 0$, we have $H(p_X) = 1$.

Entropy is a special case of cross entropy. The **cross entropy** between p_X and q_X where q_X is also some distribution over $\mathcal{X}$ is defined as

$$H(p_X, q_X) := \mathop{\mathbf{E}}_{x \sim p_X}[-\log q_X(x)] = -\sum_{x \in \mathcal{X}} p_X(x) \log q_X(x)$$

We will use the following convention about zero probabilities for cross entropy: we still treat $0 \log 0$ as zero, but we treat $C \log 0$ as $-\infty$ where $C > 0$. Clearly $H(p_X) = H(p_X, p_X)$. In fact, $H(p_X, q_X) \geq H(p_X)$ for all q_X with equality iff $q_X = p_X$.[2]

   (a) Use the linearity of expectation to show that $\mathrm{var}(X) = \mathbf{E}\left[X^2\right] - \mu_X^2$.

   (b) Let X represent the outcome of a fair six-sided die (thus $\mathcal{X} = \{1 \ldots 6\}$). Calculate the mean, variance, and entropy of p_X (round off to two decimal places).

   (c) Suppose we make the die biased so that it always yields $X = 6$. Calculate the mean, variance, and entropy of p_X (round off to two decimal places).

[1] Caution: on Google you have to type `log2` to use log base 2.

[2] This can be easily seen from the nonnegativity of the KL divergence $D_{\mathrm{KL}}(p_X || q_X) = H(p_X, q_X) - H(p_X) \geq 0$.

(d) Let p_X denote the distribution of a fair six-sided die again. Let $\mathrm{Cat}((p_i)_{i=1}^n)$ denote a categorical distribution with probabilities $p_1 \dots p_n$. Calculate

$$H\left(p_X, \mathrm{Cat}\left(\frac{1}{2}, 0, 0, 0, \frac{1}{2}, 0\right)\right) =$$
$$H\left(p_X, \mathrm{Cat}\left(\frac{1}{5}, \frac{1}{5}, \frac{1}{5}, \frac{1}{5}, \frac{1}{10}, \frac{1}{10}\right)\right) =$$
$$H\left(p_X, \mathrm{Cat}\left(\frac{1}{6}, \frac{1}{6}, \frac{1}{6}, \frac{1}{6}, \frac{1}{6}, \frac{1}{6}\right)\right) =$$

2. **(Linear Algebra)** Calculate the following matrix products, or write invalid if not possible.

$$\begin{bmatrix} 1 & -1 \end{bmatrix} \begin{bmatrix} 1 \\ 4 \end{bmatrix} = \qquad \begin{bmatrix} 1 & 4 \\ 2 & 5 \\ 3 & 6 \end{bmatrix}^\top \begin{bmatrix} 0 & 1 & 1 & -1 \\ 2 & 2 & 1 & -2 \\ 3 & 0 & 0 & 1 \end{bmatrix} =$$

$$\begin{bmatrix} 395 & -773 & 171 & 597 & 36 \\ 777 & 888 & -999 & 100 & 1 \\ -18 & -2 & -35 & 53 & 99 \\ 587 & 11 & 236 & 71 & 316 \\ 391 & 7 & 35 & 128 & 6 \end{bmatrix} \begin{bmatrix} 0 \\ 1 \\ 0 \\ 1 \\ 0 \end{bmatrix} = \qquad \begin{bmatrix} 1 & 4 \\ 2 & 5 \\ 3 & 6 \end{bmatrix} \begin{bmatrix} 0 & 1 & 1 & -1 \\ 2 & 2 & 1 & -2 \\ 3 & 0 & 0 & 1 \end{bmatrix}^\top =$$

3. **(Optimization)**

(a) Solve the following unconstrained minimization problems. Write $-\infty$ if necessary.

$$\arg\min_{x \in \mathbb{R}} \frac{1}{2}(x-3)^2 + 2 = \tag{1}$$
$$\arg\min_{x \in \mathbb{R}} \frac{1}{3}(x-3)^3 + 2 = \tag{2}$$

(b) Calculate the derivative and the second derivative of (1) and justify your answer in 3a.

(c) Calculate the derivative and the second derivative of (2) and justify your answer in 3a.

Problem 2: n-Gram Models

(4 + (2+1+1+1) = 9 points)

1. **(Relative Frequency Lemma)**: We will prove the following useful lemma:

Lemma 1. *Let $[n] = \{1 \dots n\}$ and $\Delta^{n-1} = \left\{q \in \mathbb{R}^n : q \geq 0, \sum_{i \in [n]} q_i = 1\right\}$. Let $c_i \geq 0$ be a nonnegative scalar associated with each $i \in [n]$ and assume $N = \sum_{i \in [n]} c_i > 0$. Define*

$$q^* = \arg\max_{q \in \Omega} \sum_{i \in [n]} c_i \log q_i \tag{3}$$

Then $q_i^* = c_i / N$ for each $i \in [n]$.

The Lagrangian relaxation of the constrained objective (3) is

$$\min_{\lambda \in \mathbb{R}} \max_{q \in \mathbb{R}^n} \sum_{i \in [n]} c_i \log q_i - \lambda \left(1 - \sum_{i \in [n]} q_i\right) \tag{4}$$

Intuitively, q is normalized in any finite solution of this *unconstrained* formulation because otherwise the objective is made infinitely small by taking $\lambda \to \infty$ (or $\lambda \to -\infty$). Note that we do not enforce the nonnegativity constraints because they are automatically satisfied in the maximization problem.

(a) Since $\sum_{i\in[n]} c_i \log q_i$ is concave, by the usual theory of convex programming the unique solution of (3) is given by the stationary point (q, λ) in (4) satisfying

$$\frac{\delta}{\delta\lambda} \sum_{i\in[n]} c_i \log q_i - \lambda \left(1 - \sum_{i\in[n]} q_i\right) = 0$$

$$\frac{\delta}{\delta q_j} \sum_{i\in[n]} c_i \log q_i - \lambda \left(1 - \sum_{i\in[n]} q_i\right) = 0 \qquad \forall j \in [n]$$

Solve the system and thereby prove Lemma 1.

2. **(Maximum Likelihood Estimation (MLE) of the Trigram Language Model)**: A **language model** q_X defines a distribution over possible sentences. A sentence is a finite sequence of words $x \in V^+$ where V is a finite discrete set (aka. the vocabulary). Given an unknown but samplable "true" distribution p_X over sentences, we define an MLE q_X^{MLE} as a model that maximizes the expected log likelihood, or equivalently minimizes the cross entropy between p_X and q_X,

$$q_X^{\text{MLE}} = \operatorname*{arg\,max}_{q_X\in\Theta} \mathop{\mathbf{E}}_{x\sim p_X} [\log q_X(x)] = \operatorname*{arg\,min}_{q_X\in\Theta} H(p_X, q_X) \tag{5}$$

where Θ denotes the considered model space. By the usual property of cross entropy, if $p_X \in \Theta$ then $q_X^{\text{MLE}} = p_X$. In practice, given samples $x^{(1)} \dots x^{(N)}$ drawn iid from p_X (aka. a corpus) we calculate an empirical estimate of (5) by

$$\hat{q}_X^{\text{MLE}} = \operatorname*{arg\,max}_{q_X\in\Theta} \frac{1}{N} \sum_{i=1}^{N} \log q_X\left(x^{(i)}\right) \tag{6}$$

A **trigram language model** defines for each sentence $x = (x_1 \dots x_M) \in V^M$, $M \geq 1$, the following probability

$$t(x_1 \dots x_M) = \prod_{j=1}^{M+1} t(x_j | x_{j-2}, x_{j-1}) \tag{7}$$

where $x_{-1} = x_0 = \text{BOS}$ are a special symbol denoting the beginning of a sentence, and $x_{M+1} = \text{EOS}$ is a special symbol denoting the end of a sentence (we assume this convention for a sentence of any length). The model space is $\Theta = \{t(\cdot|x, x') \in \Delta :\ x, x' \in V \cup \{\text{BOS}\}\}$ where Δ denotes a distribution over $V \cup \{\text{EOS}\}$.

(a) Given $x^{(1)} \dots x^{(N)}$ drawn iid from p_X, define for all $x, x', x'' \in V \cup \{\text{BOS}\} \cup \{\text{EOS}\}$

$$\#(x, x', x'') = \sum_{i=1}^{N} \sum_{j=1}^{|x^{(i)}|+1} \left[\left[\left(x_{j-2}^{(i)} = x\right) \wedge \left(x_{j-1}^{(i)} = x'\right) \wedge \left(x_j^{(i)} = x''\right)\right]\right]$$

where $[[S]]$ is 1 if S is true and 0 otherwise; also define $\#(x, x') = \sum_{x''} \#(x, x', x'')$. Use Lemma 1 to show that an empirical MLE (6) of the trigram language model (7) is

$$\hat{t}^{\text{MLE}}(x''|x, x') = \frac{\#(x, x', x'')}{\#(x, x')} \qquad \begin{array}{l} \forall x, x' \in V \cup \{\text{BOS}\} :\ \#(x, x') > 0 \\ \forall x'' \in V \cup \{\text{EOS}\} \end{array}$$

(b) Assume $x^{(1)}, x^{(2)}, x^{(3)} \sim p_X$ where

$x^{(1)} =$ `the dog ignored the cat`

$x^{(2)} =$ `the cat ate the mouse`

$x^{(3)} =$ `the mouse screamed`

Write all nonzero MLE parameter values $\hat{t}^{\text{MLE}}(x''|x, x')$ estimated from this corpus.

(c) What is the perplexity of the model in 2b on the corpus in 2b (in nats, i.e., log base e)?

(d) What is the perplexity of the model in 2b on the following corpus

$$x^{(1)} = \texttt{the dog ignored the cat}$$
$$x^{(2)} = \texttt{the cat ate the mouse}$$
$$x^{(3)} = \texttt{the dog screamed}$$

(in nats, i.e., log base e)?

Problem 3: Programming (2 + 1 + 1 + 1 + 2 + 1 + 3 + 1 = 12 points)

Acknowledgement. This problem is heavily adapted from A1 of COS 484 at Princeton, designed by Danqi Chen and Karthik Narasimhan.

You will experiment with a bigram language model (without BOS and EOS symbols for simplicity) with smoothing. Download the starter kit provided. The directory `data/` contains a text corpus extracted from Gigaword for training and validation. The files `assignment1.py` and `util.py` contain partially complete code. You are recommended to use Python version 3: download all necessary packages using the `pip` command and virtual environments.

1. Implement `count_ngrams` in `Tokenizer`. There is a basic test that checks for correctness.
2. Explore different choices of the tokenizer (basic, nltk, wp, bpe) by examining top-k most frequent unigrams/bigrams/trigrams (you can use `show_ngram_information`). For each choice, report the vocabulary size (using all training data and without thresholding, e.g., set `--vocab 99999999`).
3. Use the nltk tokenizer to plot the top-100 most frequent unigrams (see the `--figure` flag). Paste the resulting figure here. Does Zipf's law seem to hold (approximately)?

   For the questions below, use the nltk tokenizer and vocab size 10,000 (default).
4. Train and test the basic bigram model by running `assignment1.py`. What is the training and validation perplexity? Why do you get that validation perplexity?
5. Implement Laplace smoothing
   $$\hat{q}^{\alpha}(x'|x) = \frac{\#(x,x') + \alpha}{\#(x) + \alpha\,|V|}$$
   There is a basic test that checks for correct normalization. Plot the perplexity on both train and validation sets as a function of α (with values 10^{-5}, 10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}, 1, 10) and explain what you observe. Explain why this fixes the issue with the validation perplexity in 4.
6. Fix $\alpha = 0.01$. Vary the fraction of training data used (in increments of 10% from 10% to 100%) using the `--train_fraction` flag. Plot the train and test perplexities across this variation. Do the curves match your intuition?
7. Implement interpolation smoothing
   $$\tilde{q}^{\alpha,\beta}(x'|x) = \beta\hat{q}^{\alpha}(x'|x) + (1-\beta)\hat{q}^{\alpha}(x')$$
   where each $\hat{q}^{\alpha}$ is a Laplace-smoothed distribution. There is a basic test that checks for correct normalization. Fix $\alpha = 0.01$ and plot the perplexity on both train and validation sets as a function of β (with values in increments of 0.1 from 0.1 to 0.9) and explain what you observe.
8. Report the best validation perplexity after exploring these choices. It should not be larger than 200 nats.

   Important: For all your programming exercises, *only* add necessary parts to answer the questions without modifying anything else (e.g., the command line options) or creaing any new file. Make sure your code runs as expected before uploading. For example, the following commands should work:
   ```
   python assignment1.py
   python assignment1.py --quiet --smoothing laplace --alpha 0.01
   python assignment1.py --quiet --smoothing interpolation --alpha 0.01 --beta 0.8
   ```