CSE 291: Unsupervised learning

Homework 5

This homework is due on Thursday May 18 at 11.59pm.

- All homeworks must be typewritten and uploaded to Gradescope.
- No late homeworks will be accepted.

1. *Exponential distribution.*

   Recall that for $\alpha, \beta > 0$, the gamma(α, β) distribution over $(0, \infty)$ has density

   $$p(x) = \frac{\beta^\alpha}{\Gamma(\alpha)} x^{\alpha-1} e^{-\beta x},$$

   where $\Gamma(\cdot)$ is the gamma function.

   The gamma generalizes several well-known distributions such as the exponential and chi-squared. The **exponential distribution** with parameter $\lambda > 0$ has density

   $$p(x) = \lambda e^{-\lambda x}, \quad x \geq 0.$$

   (a) What gamma distribution is this?

   (b) What are the mean and variance of the exponential(λ) distribution?

   (c) For observations $x_1, \ldots, x_n \geq 0$, what is the maximum likelihood choice of λ?

2. *Concavity of entropy.* Show that $H(p)$ is a concave function of p. More precisely, show that for any distributions p, q over a discrete set S, and any $0 \leq \lambda \leq 1$,

   $$H(\lambda p + (1-\lambda)q) \geq \lambda H(p) + (1-\lambda)H(q).$$

   One way to do this is to express the difference of the two sides in terms of KL divergences, and to then invoke the non-negativity of KL divergence.

3. *Differential entropy.* In class we only talked about entropies of discrete distributions. For a continuous density p, the corresponding notion is the differential entropy $h(p)$, which is defined as follows:

   $$h(p) = \int p(x) \log \frac{1}{p(x)}\, dx.$$

   (a) What is the differential entropy of the uniform distribution over an interval $[a, b]$?

   (b) From part (a), you can see that differential entropy can sometimes be negative. Is it possible for discrete entropy (of a distribution defined on a finite set) to be negative?

   (c) What is the differential entropy of a normal distribution with mean μ and variance σ^2? You can use the natural logarithm, if you wish.

4. You are told that an unknown distribution over $\mathbb{R}$ has $\mathbb{E}X = 2$ and $\mathbb{E}X^2 = 10$. What is the maximum entropy distribution meeting these constraints? *Hint:* Which example from class has these features?

5. Somebody gives you a six-sided die that you initially assume to be fair: that is, you assume each of the outcomes $S = \{1, 2, \ldots, 6\}$ occurs with probability 1/6. However, this turns out not to be true.

   (a) Upon experimentation you find that

   $$\begin{aligned} \Pr(\text{the outcome is an even number}) &= 0.6 \\ \Pr(\text{the outcome is 1}) &= 0.3 \end{aligned}$$

   Find the maximum entropy distribution over S that satisfies these constraints. You should give explicit values for the probabilities of each of the six outcomes. *Hint:* Exploit symmetry to simplify the problem.

   (b) Repeat the same problem, for a different set of constraints:

   $$\begin{aligned} \Pr(\text{the outcome is an even number}) &= 0.5 \\ \Pr(\text{the outcome is} < 3) &= 0.5 \end{aligned}$$

6. *Text generation using n-grams.* In this question, we will try to emulate the language of Shakespeare using an n-gram language model. We will base the model on a collection of Shakespeare's sonnets and plays: the text and an accompanying Jupyter notebook are provided on the course webpage, in `text-gen.zip`.

   Take a careful look through the Jupyter notebook. In the implementation, `<START>` tokens are appended at the beginning of each sentence as indicators. A fullstop (period) token is used to indicate the end of a sentence.

   (a) Complete the `generate_n_grams` method in class `NGramModel`.
   This function creates a list of n-grams from the tokenized text. An n-gram is a tuple of $n-1$ context tokens followed by the current token. Make sure the data type of your output matches with the return data type of the function (specified after a right arrow); Python does not enforce data type check but you will still get a run time error later in the code if it doesn't match.

   (b) Complete the `get_prob` method in class `NGramModel`.
   This function should return the probability of a token given the context tokens. Each context can be associated with multiple tokens in an n-gram model. For example in a trigram ($n = 3$) model {`(Cats, are): cute, evil, pets`}, `"cute"`, `"evil"`, and `"pets"` are associated with the same context `(Cats, are)`.
   `get_prob(context, target)` should return the probability:

   $$P(\textbf{target}|\textbf{context}) = \frac{N(\textbf{context and target})}{N(\textbf{context})} \tag{5.1}$$

   In the above $N(x)$ denotes the total count of x.

   (c) Create n-gram models for $n = 2, 3$ and, for each of these values of n, generate a `100`-word text using the trained model.

   (d) Complete the function `model_ll`, which takes a sentence and an n-gram model, and returns the log likelihood of the sentence under that model. Use this function to get the log-likelihood of the provided sentence under unigram and bigram models.

(e) Given a prompt from one of Shakespeare's sonnets, generate the next 20 words using the bigram and trigram models, and report the two texts.

Submission format: For parts (a), (b), and (d), screenshot and crop your function implementation and assign it to the respective section on gradescope. Only the function code should be attached.

For parts (c) and (e), show the texts that are generated.

For part (d), give the log-likelihood of the sentence under the two models.