Homework 4, Machine Learning, Fall 2022

***IMPORTANT* Homework Submission Instructions**

1. All homeworks must be submitted in one PDF file to Gradescope.
2. Please make sure to select the corresponding HW pages on Gradescope for each question.
3. For all theory problems, please type the answer and proof using Latex or Markdown, and export the tex or markdown into a PDF file.
4. For all coding components, complete the solutions with a Jupyter notebook/Google Colab, and export the notebook (including both code and outputs) into a PDF file. Concatenate the theory solutions PDF file with the coding solutions PDF file into one PDF file which you will submit.
5. Failure to adhere to the above submission format may result in penalties.

All homework assignments must be your independent work product, no collaboration is allowed. You may check your assignment with others or the internet after you have done it, but you must actually do it by yourself. **Please copy the following statement at the top of your assignment file**:

Agreement: This assignment represents my own work. I did not work on this assignment with others. All coding was done by myself.

1 VC dimension of Binary Decision Trees with Fixed Split Points

This problem is much easier than the one from the last homework! In this problem we consider the task of binary classification with labels $\{-1, 1\}$.

Recall that binary decision trees work by splitting on points in the feature space. Those points are called split points. Let $\mathcal{X}$ be a set of fixed, predetermined and distinct split points on the domain X.

We consider the following set of binary decision trees:

$$\mathcal{F} := \{\text{the set of all binary decision trees whose split points are exactly equal to } \mathcal{X} \text{ with exactly } \ell \text{ leaves}\}$$

In other words, $\mathcal{F}$ is a set of trees that are identical except for the choice of predicted labels assigned to each leaf.

What is the VC dimension of $\mathcal{F}$?

2 Topic Modeling with EM (Ed Tam)

In this question, we explore how to use the EM algorithm to perform inference for probabilistic machine learning models.

The goal of the machine learning model in this question is to learn the latent "topics" within a collection of documents. We set up the problem in the following way:

- We have a word dictionary, $\mathcal{W}$, with a size of N words. $\mathcal{W}$ is a list of all unique words that appear at least once in our documents. We use n to index the N words.
- We have a set of M documents, $d_1, \cdots, d_M$ each represented as a $N \times 1$ column vector, with the ith entry representing the number of times the ith word in the dictionary appeared in the document. (We're using a "bag of words" representation for each document, where a document is only represented by the collection of words it contains and not the order of the words.)
- We can use the notation $q(w_n; d_i)$ for the number of times the word w_n appears in document d_i.
- We consider a set of K topics $z_1, \cdots, z_k, \cdots, z_K$. We define each topic z as a probability distribution over the words in the dictionary $\mathcal{W}$. E.g., you can imagine a topic that represents, say, "sports" would put higher probability on words like "basketball," "practice," "score," etc., whereas a topic that represents "music" would put higher probability on words like "guitar," "piano," "chords," etc. (but interestingly, the word "score" might also come up in music).
- We think of each document as being composed by a combination of topics. For example, a given document might be 70 percent on the topic "sports," 20 percent on the topic "food," and 10 percent on the topic "science." Of course, these percentages/proportions of topics must sum up to 1. In ML/statistics speak, each document is generated by a "mixture" of topics.

We imagine the collection of words in each of the documents that we have collected as independently generated by some underlying probabilistic model. A reasonable model to think about the generative process for the words in each document is as follows:

- Recall that each topic is a distribution on $\mathcal{W}$, the set of words. Mathematically, we model the probability that word n appears in topic k as

$$\Pr(w_n | z_k) = \beta_{kn}, \quad \sum_{n=1}^{N} \beta_{kn} = 1$$

  In other words, the conditional distribution of a word given a topic is a categorical distribution.

- Similarly, documents are composed by combinations of various topics, with each document having a different distribution/weighting on each topic. Mathematically, we model the proportion of a topic z_k within a document d_i as

$$\Pr(z_k|d_i) = \alpha_{ik}, \quad \sum_{k=1}^{K} \alpha_{ik} = 1$$

  In other words, the conditional distribution of a topic given a document is also a categorical distribution.

- To ease notation, when we are referring to the collection of all the $\{\beta_{kn}\}_{k=1,n=1}^{k=K,n=N}$ parameters, we use the notation β. Similarly, we use α to denote the collection of all $\{\alpha_{ik}\}_{i=1,k=1}^{i=M,k=K}$ parameters.

We are interested in inferring α_{ik} and β_{kn} because they're informative: α_{ik} represents the proportion of topic k that makes up document i and β_{kn} represents the likelihood of seeing word n under topic k.

As a result, in a document d_i, a single word w_n is formed as follows:

1. Sample a topic z_k from the topic distribution $\Pr(z_k|d_i)$ (with probability α_{ik})
2. Sample a word (from the sampled topic in step 1) according to the vocabulary distribution $\Pr(w_n|z_k)$ (with probability β_{kn})

Each word is sampled independently according to the two sampling steps above.

We denote the probability p(word n appears in document i$|\alpha, \beta$) as p_{ni} for simplicity. We can conclude from the description of the model that $p_{ni} = \sum_{k=1}^{K} \Pr(w_n|z_k)\Pr(z_k|d_i) = \sum_{k=1}^{K} \beta_{kn}\alpha_{ik}$ (this is just the law of total probability. Word n can come from topic z_1, or z_2, or z_K ... if we sum up all the possibilities, that gives us the total probability.)

Our goal is to learn these parameters (the α's and the β's) by utilizing the Expectation-Maximization (EM) algorithm to maximize the log-likelihood function. The way the EM algorithm works is it iteratively updates its estimates of α and β via repeating two steps: the E-step and the M-step.

2.1 Derive Log-Likelihood

In this subquestion, we want you to write down and simplify the log-likelihood of the above topic model.

Hint: if you don't know where to start, think about the probability of word n appearing in document i.

Your answer should involve α, β and q.

2.2 E Step

Now that we have derived the form of the log-likelihood of our model, what we want is to find values of α and β to maximize the log-likelihood. The EM algorithm is one way to achieve this goal.

The EM algorithm is an iterative algorithm, which means that we will update the values of α and β iteratively. We use the notation $\alpha^{old}, \beta^{old}$ to denote the values of α and β at the previous iteration, and $\alpha^{new}, \beta^{new}$ to denote the values of α and β at the next iteration.

In the E step, we want to derive an expression for the proportion of topic z_k in document d_i, given that word n is in document d_i. Concretely, we want to find an expression for $p(z_k|d_i, w_n, \alpha^{old}, \beta^{old})$. Note that this probability is for a specific k, n and i. Also note that your expression should only depend on $\alpha^{old}, \beta^{old}$.

2.3 Find ELBO for M-Step

After the E-step, we want to proceed to the M-Step, which finds α and β values that maximizes a lower bound for the log-likelihood.

This lower bound of the log-likelihood is often called the Evidence Lower Bound (ELBO), which we will denote as $A(\alpha, \beta)$. Note that it is a function of α and β.

To ease notation, define $\gamma_{ink} := p(z_k|w_n, d_i, \alpha^{old}, \beta^{old})$ (note that this is what we found in the E-step.)

Show that the following function $A(\alpha, \beta)$ is a lower bound for the log-likelihood.

$$A(\alpha, \beta) = \sum_{i=1}^{M} \sum_{n=1}^{N} \sum_{k=1}^{K} q(w_n, d_i) \gamma_{ink} \log \left(\frac{\alpha_{ik} \beta_{kn}}{\gamma_{ink}} \right).$$

2.4 M Step

Now that we have the ELBO, in the M-step we want to find the optimal values of α and β that maximizes the ELBO. We will assign the optimal values to $\alpha^{new}, \beta^{new}$. Find the form of $\alpha^{new}, \beta^{new}$.

Hint: Don't forget the constraints on α and β.

3 Gradient computations in Neural Networks (Ed Tam)

Suppose you have a feedforward neural network with architecture $784 - H - H - 1$, where H is a positive integer that represent the number of hidden nodes in the middle layers. The number of hidden layers in this model is 2. The model can be represented as

$$f(\mathbf{x}) : \mathbb{R}^{784} \to [0, 1]$$

where the learnable parameters are $\{\mathbf{W}_1 \in \mathbb{R}^{784 \times H}, \mathbf{b}_1 \in \mathbb{R}^H, \mathbf{W}_2 \in \mathbb{R}^{H \times H}, \mathbf{b}_2 \in \mathbb{R}^H, \mathbf{w}_3 \in \mathbb{R}^H, b_3 \in \mathbb{R}\}$. Explicitly, $f(\mathbf{x})$ is

$$\begin{aligned} \mathbf{h}_1 &= \sigma\left(\mathbf{W}_1^\top \mathbf{x} + \mathbf{b}_1\right) \\ \mathbf{h}_2 &= \sigma\left(\mathbf{W}_2^\top \mathbf{h}_1 + \mathbf{b}_2\right) \\ f(\mathbf{x}) = \mathbf{h}_3 &= \sigma\left(\mathbf{W}_3^\top \mathbf{h}_2 + b_3\right) \end{aligned}$$

where the activation function is $\sigma(z) = \frac{1}{1+e^{-z}}$. Recall that the activation function is applied element-wise, which means if $\mathbf{x} = (x^1, x^2, ..., x^d) \in \mathbb{R}^d$, we have $\sigma(\mathbf{x}) = (\sigma(x^1), \sigma(x^2), ..., \sigma(x^d))$.

The loss function for this model (or the negative log-likelihood) when doing binary classification with labels $y \in \{0, 1\}$ is the usual one:

$$\mathcal{L} = -\sum_{i=1}^{N} y_i \log(f(\mathbf{x}_i)) + (1 - y_i) \log(1 - f(\mathbf{x}_i)).$$

Here, $\{(\boldsymbol{x}_i, y_i)\}_{i=1}^N$ is a training sample.

Some useful notation that might come in handy is to define $\boldsymbol{z}_l := \mathbf{W}_l^\top \boldsymbol{h}_{l-1} + \boldsymbol{b}_l$ and to define $\delta_l^i := \frac{\partial \mathcal{L}_i}{\partial \boldsymbol{z}_l}$. In other words, $\boldsymbol{z}_l$ is the "pre-activation" at layer l.

3.1 Gradient Evaluation

Derive the gradient of $\frac{\partial \mathcal{L}}{\partial \mathbf{W}_1}$ by backpropagation.

Remark: One way to approach this problem is to think about the gradient of the loss function $\mathcal{L}_i$ for each training data point i first, and then add the $\mathcal{L}_i$'s up later to obtain $\mathcal{L}$. (Of course, you do not necessarily have to follow this recipe, as there are other ways to solve this problem.)

3.2 General Rule

Based on the above derivations for 3.1, in the case where there are $L-1$ hidden layers in the neural network, deduce a general rule for computing $\delta_l^i := \frac{\partial \mathcal{L}_i}{\partial \boldsymbol{z}_j}$ in terms of δ_{l+1}^i, $\mathbf{W}_{l+1}$ and $\mathbf{h}_l$ for any integer layer number l that lies within $L-1 \geq l \geq 1$.

4 Clustering (Yi)

In this problem, we will explore several clustering algorithms using mall_customers dataset. This dataset consists of 200 samples with two features: annual income and spending score.

1. Load the dataset and draw a scatter plot of the dataset.

2. Based on the distribution of the data points, select a reasonable k and implement the k-means algorithm in the language of your choice, using the euclidean distance (between each data point and the cluster center) as the distance metric. The **arguments** of your function are k and the input dataset. **Return** both the cluster assignment for each point in the dataset, as well as the center for each cluster. You can initialize each center by randomly selecting a point from the input data. (Note that it is generally a good idea to perform multiple replicates for k-means.)

   Then draw a scatter plot of the dataset, using different colors for different clusters. Mark the center of each cluster. Intuitively, what kind of customers does each cluster represent?

3. We can also directly use *KMeans* in *sklearn*. Choose the same number of clusters as in the previous question. Do you get **exactly** the same results when using *sklearn* and your implementation? Calculate which one is better.

4. K-medians is a variation of k-means, which minimizes error over all clusters with respect to the 1-norm distance metric. Implement the k-medians algorithm from scratch, using the same k in the previous questions. The **arguments** of your function are k and the input dataset. **Return** both the cluster assignment for each point in the dataset, as well as the median for each cluster.

   Then draw a scatter plot of the dataset, using different colors for different clusters. Mark the median of each cluster.

   Do you get **exactly** the same results when using k-means and k-medians?

5 Convolutional Neural Network on CIFAR-10 (Yi)

In this problem, we will train a convolutional neural network to classify the CIFAR-10 dataset. The dataset consists of 60000 32x32 images in 10 classes, with 6000 images per class. There are 50000 training images and 10000 test images. We will not need to use a GPU for this problem. With a CPU, training the network for 30 epochs takes roughly 10 to 15 minutes. We will be using PyTorch for this problem. If you need help there are many online resources including the official Pytorch documentation: https://pytorch.org/tutorials/beginner/blitz/neural_networks_tutorial.html

1. Follow the skeleton code in ***cifar10.ipynb*** and fill in the holes in the code (marked with "TODO").

2. Plot the training losses and validation losses in one figure, and use another figure to plot the training accuracies and validation accuracies. Is this model overfitting?

3. Report the test accuracy, and show the confusion matrix for test. Which class seems to get confused the most? Intuitively, why does that make sense?