The Australian National University
School of Computing
Conveners: Thang Bui & Jing Jiang

Semester 1, 2025
Assignment 2
Weight: 25%

COMP4670/8600: Statistical Machine Learning

Release Date. 22nd April 2025.
Due Date. 25th May 2025 at 23:59 AEST.
Maximum credit. 100 Marks.

Neural Networks, Mixture Models, and Gibbs Sampling

In this assignment, you will have the opportunity to review and have hands-on practice on some of the topics covered in the second half of the semester. The first two questions of the assignment are related to neural networks. The third question uses a mixture model and employs the EM algorithm to uncover the main topics from a set of news articles. The fourth question extends the third question and uses Gibbs sampling to uncover the topics.

For submission Name your answer file as `uIDNUMBER.pdf`, add the file together with the *three* python files that you have edited (i.e., `a2_nn.py`, `a2_mixture.py`, `a2_sampling.py`) into a zip file, name it as `uIDNUMBER.zip`, and then submit the zip file on Wattle. Ensure that the coding solutions are in the designated files (as specified). Do not include additional files, such as your locally installed packages, additional figures, or the data files.

Coding The code has been tested with standard packages such as `numpy` and `matplotlib`. You do not need additional packages for this assignment.

Collaboration You are free to discuss the material in the assignment with your classmates. However, every answer and code submitted must be written *by yourself without assistance*. You should be able to explain your answers if requested.

Late submission policy We allow a 5-minute grace period after midnight. Assignment submissions that are late from 5 minutes to 24 hours attract a 5% penalty (of possible marks available). Submissions late by more than 24 hours without an approved extension will get zero. Please submit early to avoid potential connection issues.

Extension requests will be processed via the Extension online form available on the CSS website.

Regrade requests will be processed via a Microsoft Form. This will be released in due course.

Other notes:

- When writing proofs, use the equation numbers when referring the equations in the assignment, *i.e.*, "Through Equation (3.1) we can show ...".
- You are not required to complete the assignment linearly, you may want to solve whichever questions you can regardless of order of appearance.
- Unless stated otherwise, code is only graded on the correctness of functions implemented, not on performance or code quality. Our main requirement on performance/code quality is that we can run your code in a reasonable time when marking it.

Section 1: Backpropagation (18 Points)

In the lectures and in Lab 5, we have seen how backpropagation works for a full-connected (a.k.a. feedforward) neural network. In this question, we will look at a simple non-fully-connected neural network and work out its backpropagation process. You will NOT need to write any code for this question.

Figure 1 shows a neural network with two hidden layers. The first hidden layer has two hidden nodes, $z_1^{(1)}$ and $z_2^{(1)}$, and the second hidden layer has three hidden nodes, $z_1^{(2)}$, $z_2^{(2)}$ and $z_3^{(2)}$. Note that this neural network is not fully connected.

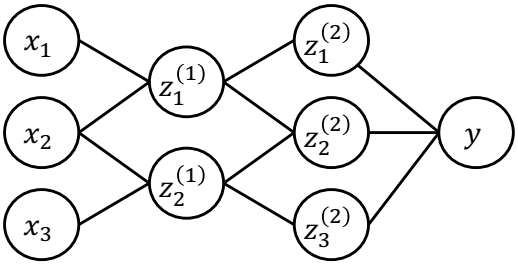

Figure 1: A neural network with two hidden layers.

Suppose we are using this neural network for a binary classification problem. The output node y uses the sigmoid function as shown below as its activation function.

$$\sigma(x) = \frac{1}{1+e^{-x}}. \tag{1.1}$$

Let $t \in \{0, 1\}$ be a binary variable that denotes the final class label. The output value y given input values x_1, x_2, x_3 is interpreted as the probably $p(t = 1|x_1, x_2, x_3)$.

Let us further assume that the two hidden layers use the tanh function as their activation functions, which is defined below:

$$\tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}. \tag{1.2}$$

Let $w_{jk}^{(l)}$ denote the weight between the nodes $z_j^{(l-1)}$ and z_k^l, where the node $z_j^{(0)}$ is defined to be x_j and the node $z_1^{(3)}$ is y.

Question 1.1: Parameters of the model (2 Points)

How many parameters (including both weights and bias parameters) does this neural network model have? Show your final answer as an integer and explain what these parameters are, i.e., how you have arrived at your final answer.

Let us assume that we have a set of N training data denoted as $\{x_{i1}, x_{i2}, x_{i3}, t_i\}_{i=1}^N$, where $t_i \in \{0, 1\}$.

Let us use the cross-entropy loss over the entire training set as the loss function:

$$\mathcal{L} = -\sum_{i=1}^{N} \Big(t_i \log p(t = 1|x_{i1}, x_{i2}, x_{i3}) + (1 - t_i) \log p(t = 0|x_{i1}, x_{i2}, x_{i3}) \Big) \tag{1.3}$$

For all the questions below, you can assume that a forward pass has taken place and all the nodes have their output values computed and stored. You can use $z_{ij}^{(l)}$ to denote the output value of the node $z_j^{(l)}$ for the i-th data point. Similarly, you can use y_i to denote the output value of the node y for the i-th data point.

For all the questions below, you need to show the step-by-step derivation of your final answers.

Question 1.2: $\frac{\partial \mathcal{L}}{\partial w_{11}^{(3)}}$ (4 Points)

With the definition of $\mathcal{L}$ in Eqn. 1.3, show how $\frac{\partial \mathcal{L}}{\partial w_{11}^{(3)}}$ is expressed as a function in terms of the t_i's, the output values of the various nodes (y_i's, $z_{i1}^{(2)}$'s, etc.), and the values of the weight parameters (i.e., $w_{jk}^{(l)}$'s).

Question 1.3: $\frac{\partial \mathcal{L}}{\partial w_{22}^{(2)}}$ (4 Points)

With the definition of $\mathcal{L}$ in Eqn. 1.3, show how $\frac{\partial \mathcal{L}}{\partial w_{22}^{(2)}}$ is expressed as a function in terms of the t_i's, the output values of the various nodes (y_i's, $z_{i1}^{(2)}$'s, etc.), and the values of the weight parameters (i.e., $w_{jk}^{(l)}$'s).

Question 1.4: $\frac{\partial \mathcal{L}}{\partial w_{21}^{(1)}}$ (6 Points)

With the definition of $\mathcal{L}$ in Eqn. 1.3, show how $\frac{\partial \mathcal{L}}{\partial w_{21}^{(1)}}$ is expressed as a function in terms of the t_i's, the output values of the various nodes (y_i's, $z_{i1}^{(2)}$'s, etc.), and the values of the weight parameters (i.e., $w_{jk}^{(l)}$'s).

Question 1.5: Backpropagation (2 Points)

Using the computation of $\frac{\partial \mathcal{L}}{\partial w_{21}^{(1)}}$ in the network above as an example, explain why in general the computation of $\frac{\partial \mathcal{L}}{\partial w_{jk}^{(l)}}$ should start from the last layer rather than the first layer to save computational costs.

Section 2: Neural Network for Regression (22 Points)

In this question, we will build a simple fully-connected neural network for two simple regression problems.

Besides σ and tanh, another commonly used activation function is the Rectified Linear Unit (ReLU) function, as defined below:

$$\text{ReLU}(x) = \begin{cases} 0 & \text{if } x < 0 \\ x & \text{if } x \geq 0 \end{cases}. \tag{2.1}$$

Question 2.1: ReLU (5 Points)

Inside the file `a2_nn.py` provided to you, implement the class `ReLU`. (Hint: Refer to the implementation of the class `Sigmoid` in Lab 5 as an example.)

For our regression problems, let us use the mean squared error as our loss function, which is defined as follows:

$$\mathcal{L} = \frac{1}{N} \sum_{i=1}^{N} (t_i - y_i)^2, \tag{2.2}$$

where t_i is the observed output value given input x_i and y_i is the predicted output value based on the neural network, and we assume that there are in total N data points in the training dataset.

Question 2.2: Squared error loss function (5 Points)

Inside the file `a2_nn.py` provided to you, implement the class `SquaredErrorLoss` based on the loss function shown in Eqn. 2.2. (Hint: Refer to the implementation of the class `BinaryCrossEntropyLoss` in Lab 5 as an example.)

We will use a fully-connected neural network as shown in Figure 2 for this question. This network has three hidden layers, where the first hidden layer has `P` hidden nodes, the second hidden layer has `Q` hidden nodes, and the third (last) hidden layer has exactly 2 hidden nodes. The activation function to be used is ReLU for all hidden nodes. The input layer has a single node, and the output layer also has a single node. The output node does not use any activation function.

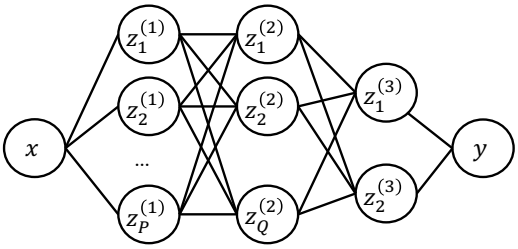

Figure 2: A fully-connected neural network with three hidden layers.

Question 2.3: Building the neural network (3 Points)

Inside the file `a2_nn.py` provided to you, implement the class `Network`. `P` and `Q` are parameters to be pass to the network when it is instantiated. (Hint: You can re-use some of the code from Lab 5. You can refer to the class `Network` in Lab 5 as an example.)

Next, let us consider two regression tasks. In the first regression task, we try to use the neural network to fit the curve $y = x^2$, whereas in the second one, we try to use the neural network to fit the curve $y = \text{abs}(x)$, i.e., the absolute value of x.

You have been given the starting code inside `a2_nn.py` that creates a series of (x, y) pairs, stored in `x` and `y`. Note that you will need to uncomment either the line `y = x ** 2` or the line `y = np.abs(x)` to create the y values. The data has been split into a training set and a test set.

Question 2.4: Training the neural network (6 Points)

Write code in `a2_nn.py` to train a neural network with each of the following configurations for each of the two regression tasks, using the training data `(x_train, y_train)`. Each time train the network for 1000 epochs.

1. `P = 2, Q = 2`
2. `P = 5, Q = 5`
3. `P = 10, Q = 10`

For each regression task and for each network configuration above, generate the following two plots:

- A plot with two curves showing the training loss and test loss over the 1000 epochs, respectively.
- A plot with two curves showing the actual output value of y and the predicted output value of y over the input value x, for all the x in the test dataset (i.e., inside `x_test`).

Include these plots in your report.

Question 2.5: Visualising the hidden nodes in the last hidden layer (3 Points)

The last hidden layer of your network always has exactly 2 hidden nodes. Pick one of the configurations where your network has fit the curve $y = \text{abs}(x)$ well. Plot the output values of the two hidden nodes $z_1^{(3)}$ and $z_2^{(3)}$ over the input value x. Label the two curves as `z(3)_1` and `z(3)_2`, respectively. Include this plot in your report.

Based on your plot, what do you expect the two weights connecting $z_1^{(3)}$ to y and connecting $z_2^{(3)}$ to y to be? Print out the values of the two weights and discuss whether your guess roughly matches the actual values of these two weights.

Section 3: Mixture Models and the EM Algorithm (35 Points)

In the lectures, we have discussed mixtures of Gaussians and how the EM algorithm can be used to estimate the parameters of each mixture component as well as the mixing coefficients. In this question, we will use a mixture model to uncover the "hidden topics" in a set of news articles.

You are given a data file called `articles.txt` that contains a set of news articles.[1] Each line of the file contains a single news article, where the first column is the body of the article, the second column is the headline of the article, and the third column is the category of the article. The three columns are separated by tabs. For this question, we will use only the first two columns of the file, i.e., the body and the headline of each article.

We will represent each news article as two sequences of words: $(w^b_{i,1}, w^b_{i,2}, \ldots, w^b_{i,M_i})$ are the words found in the body of the i-th article, where M_i is the number of words in the i-th article's body, and $(w^h_{i,1}, w^h_{i,2}, \ldots, w^h_{i,L_i})$ are the words found in the headline of the i-th article, where L_i is the number of words in the i-th article's headline. We ignore words that appear frequently in many articles but are usually not representative of any topic, such as "a", "an", "the", "and", "or", "to", and "of". These words are called "stopwords" in language processing. For convenience, after stopwords are removed, each remaining word is mapped to an ID (which is an integer) ranging from 1 to V, where V is the vocabulary size (i.e., the number of unique words in the dataset). Note that we have provided you with the pre-processing code, so you do not need to perform stopword removal or the mapping from words to the word IDs. For the rest of the discussion, we will assume that $w^b_{i,j}$'s and $w^h_{i,j}$'s are word IDs, i.e., integers in $[1, V]$.

We assume that the words in the body of an article are generated from a *mixture* of K categorical distributions (a.k.a. generalised Bernoulli distributions or multinoulli distributions).[2] Similarly, the words in the headline of an article are also generated from a mixture of (a different set of) K categorical distributions. Let us use $\{\boldsymbol{\theta}^b_k\}_{k=1}^K$ to denote the parameters of the K categorical distributions for the article bodies, where $\boldsymbol{\theta}^b_k$ is a categorical distribution over V outcomes. Specifically, $\boldsymbol{\theta}^b_k$ is a V-dimensional vector where $\theta^b_{k,v} \in [0, 1]$ is the probability of selecting the word with the ID v ($1 \leq v \leq V$) from this categorical distribution. Similarly, we will use $\{\boldsymbol{\theta}^h_k\}_{k=1}^K$ to denote the parameters of the K categorical distributions for the article headlines. These categorical distributions over the vocabulary are commonly referred to as "topics". For the rest of this question, we will refer to $\{\boldsymbol{\theta}^b_k\}_{k=1}^K$ as the K "body topics" and $\{\boldsymbol{\theta}^h_k\}_{k=1}^K$ as the K "headline topics".

The mixing coefficients are different for different articles. Specifically, we will use $\boldsymbol{\pi}_i$, a K-dimensional vector, to denote the mixing coefficients for the i-th article, where $\pi_{i,k} \in [0, 1]$ denotes the probability to select the k-th body (or headline) topic among the K body (or headline) topics. $\boldsymbol{\pi}_i$'s are essentially the parameters of a categorical distribution over K outcomes.

We assume that for each word $w^b_{i,j}$ in the body of the i-th article, there is a hidden variable $z^b_{i,j}$ associated with it. $z^b_{i,j}$ is a discrete variable that takes a value between 1 and K, denoting the topic chosen for that word. $z^b_{i,j}$ follows the categorical distribution parameterised by $\boldsymbol{\pi}_i$. Given a value of $z^b_{i,j}$, the word $w^b_{i,j}$ follows the categorical distribution parameterised by $\boldsymbol{\theta}^b_{z^b_{i,j}}$.

Similarly, for each word $w^h_{i,j}$ in the headline of the i-th article, there is a corresponding hidden variable $z^h_{i,j}$, which also follows the categorical distribution parameterised by $\boldsymbol{\pi}_i$. Given a value of $z^h_{i,j}$, the word $w^h_{i,j}$ follows the categorical distribution parameterised by $\boldsymbol{\theta}^h_{z^h_{i,j}}$.

Question 3.1: The log likelihood function (5 Points)

Based on the description of the mixture model above, give the formula of the log likelihood of the N observed news articles $\{((w^b_{i,1}, w^b_{i,2}, \ldots, w^b_{i,M_i}), (w^h_{i,1}, w^h_{i,2}, \ldots, w^h_{i,L_i}))\}_{i=1}^N$. Show the derivation of your formula. (Hint: You will need to sum over all possible values of each $z^b_{i,j}$ and each $z^h_{i,j}$ when deriving the log likelihood function.).

Question 3.2: The E-step and the M-step (10 Points)

Derive the EM algorithm for the mixture model above. Show the derivation steps of your final answers.

Question 3.3: Implementation (10 Points)

Based on the starting code provided in `a2_mixture.py`, implement the EM algorithm.

Note: If you find that some articles have no words left in either the body or the headline after pre-processing, you can discard those articles. We will only examine the discovered topics when analysing the results.

[1] The data comes from this public dataset: `https://www.kaggle.com/datasets/timilsinabimal/newsarticlecategories`.

[2] See `https://en.wikipedia.org/wiki/Categorical_distribution`.

Question 3.4: Results (6 Points)

Set K to 5, 10, and 20. Run the EM algorithm. For each configuration, show the top-10 words with the highest probabilities in each $\boldsymbol{\theta}_k^b$ and each $\boldsymbol{\theta}_k^h$.

Question 3.5: Discussion (4 Points)

Discuss the results you have observed by answering the following three questions:

1. For each setting, do you observe meaningful topics?
2. Which setting ($K = 5$, 10, or 20) gives the most meaningful topics, in your opinion?
3. Did you expect to see alignment (i.e., similar top-ranked words) between $\boldsymbol{\theta}_k^b$ and $\boldsymbol{\theta}_k^h$ for each k? Why?

Section 4: Gibbs Sampling (25 Points)

For the model described in the previous question, let us place prior distributions over the model parameters $\boldsymbol{\pi}$'s and $\boldsymbol{\theta}$'s. Specifically, we assume that each categorical distribution $\boldsymbol{\pi}_i$ is sampled from a symmetric Dirichlet distribution parameterised by α (denoted as $\text{Dir}(\alpha)$), each categorical distribution $\boldsymbol{\theta}_k^b$ is sampled from a symmetric Dirichlet distribution parameterised by β_1 (denoted as $\text{Dir}(\beta_1)$), and each categorical distribution $\boldsymbol{\theta}_k^h$ is sampled from a symmetric Dirichlet distribution parameterised by β_2 (denoted as $\text{Dir}(\beta_2)$).

Let $\boldsymbol{w}^b$ denote all the words in the bodies of the articles and $\boldsymbol{w}^h$ denote all the words in the headlines of the articles. Similarly, let $\boldsymbol{z}^b$ denote all the hidden variables associated with the words in the bodies of the articles, and $\boldsymbol{z}^h$ the hidden variables associated with the words in the headlines of the articles.

A common technique to estimate the model parameters $\boldsymbol{\pi}$'s and $\boldsymbol{\theta}$'s with the model described above is to first use collapsed Gibbs sampling to sample the hidden topics assigned to each word, i.e., $\boldsymbol{z}^b$ and $\boldsymbol{z}^h$. Given the sampled $\boldsymbol{z}^b$ and $\boldsymbol{z}^h$, the parameters $\boldsymbol{\pi}$'s and $\boldsymbol{\theta}$'s can be relatively easily estimated.

Collapsed Gibbs sampling attempts to obtain a sample of $\boldsymbol{z}^b$ and $\boldsymbol{z}^h$ based on the following posterior distribution:

$$p(\boldsymbol{z}^b, \boldsymbol{z}^h | \boldsymbol{w}^b, \boldsymbol{w}^h, \alpha, \beta_1, \beta_2).$$

Question 4.1: The collapsed Gibbs sampling formulas (7 Points)

Derive the collapsed Gibbs sampling formulas, i.e., $p(z_{ij}^b = k | \boldsymbol{z}_{\neg ij}^b, \boldsymbol{z}^h, \boldsymbol{w}^b, \boldsymbol{w}^h, \alpha, \beta_1, \beta_2)$ and $p(z_{ij}^h = k | \boldsymbol{z}^b, \boldsymbol{z}_{\neg ij}^h, \boldsymbol{w}^b, \boldsymbol{w}^h, \alpha, \beta_1, \beta_2)$. Here $\boldsymbol{z}_{\neg ij}^b$ denotes all $\boldsymbol{z}^b$ except z_{ij}^b. $\boldsymbol{z}_{\neg ij}^h$ is similarly defined.

Hint: For Question 4.1, you can use the following properties of Dirichlet-multinomial distributions without having to derive it yourself.[3]

Let $(x_1, x_2, \ldots, x_N)$ denote a sequence discrete variables identically and independently drawn from a categorical distribution with K outcomes, parameterised by $\mathbf{p}$. That is, x_n $(1 \leq n \leq N)$ is an integer between 1 and K, and p_k is the probability that x_n takes the value of k. Let $\mathbf{c} = (c_1, c_2, \ldots, c_K)$ denote the category counts, i.e., c_k is the number of x_n's $(1 \leq n \leq N)$ that has a value of k. Assume that $\mathbf{p}$ follows a Dirichlet distribution parameterised by $\boldsymbol{\alpha}$. Then we have

$$\begin{aligned} p(\mathbf{x}|\boldsymbol{\alpha}) &= \int_{\mathbf{p}} p(\mathbf{x}|\mathbf{p}) p(\mathbf{p}|\boldsymbol{\alpha})\, d\mathbf{p} \\ &= \frac{\Gamma(\alpha_0)\Gamma(N+1)}{\Gamma(\alpha_0+N)} \prod_{k=1}^{K} \frac{\Gamma(\alpha_k + c_k)}{\Gamma(\alpha_k)\Gamma(c_k+1)}, \end{aligned}$$

where $\alpha_0 = \sum_{k=1}^{K} \alpha_k$, and $\Gamma(\cdot)$ is the gamma function. Note that in this question, we use symmetric Dirichlet priors, which means all α_k's are equal. The following property of the gamma function can be directly used in your derivation:

$$\Gamma(t+1) = t\Gamma(t).$$

[3] See `https://en.wikipedia.org/wiki/Dirichlet-multinomial_distribution`.

Then derive how model parameters $\boldsymbol{\pi}$'s and $\boldsymbol{\theta}$'s can be estimated from a sample $\boldsymbol{z}^b$ and $\boldsymbol{z}^h$.

Question 4.2: Estimation of $\boldsymbol{\pi}$'s, $\boldsymbol{\theta}^b$'s and $\boldsymbol{\theta}^h$'s (3 Points)

Given a particular sample of $\boldsymbol{z}^b$ and $\boldsymbol{z}^h$, derive the formulas for estimating the model parameters $\boldsymbol{\pi}$'s, $\boldsymbol{\theta}^b$'s and $\boldsymbol{\theta}^h$'s.

Hint: You are essentially estimating the parameters based on the following posterior distribution:

$$p(\boldsymbol{\pi}, \boldsymbol{\theta}^b, \boldsymbol{\theta}^h | \boldsymbol{z}^b, \boldsymbol{z}^h, \boldsymbol{w}^b, \boldsymbol{w}^h, \alpha, \beta_1, \beta_2).$$

You can use the property we discussed in the lecture that the posterior distribution of a categorical distribution with a Dirichlet prior is still a Dirichlet distribution. You can estimate $\boldsymbol{\pi}$, $\boldsymbol{\theta}^b$, and $\boldsymbol{\theta}^h$ using either maximum a posterior estimation (i.e., choosing the one that maximises the posterior distribution) or using the expected values of $\boldsymbol{\pi}$, $\boldsymbol{\theta}^b$, and $\boldsymbol{\theta}^h$ under the posterior distribution. For either one, you can directly use the formula for the mode or the mean of a Dirichlet distribution.

Question 4.3: Implementing the Gibb sampler (10 Points)

Create a file called `a2_sampling.py`. You can re-use the relevant code from `a2_mixture.py` to pre-process the data. Implement the collapsed Gibbs sampler as defined above.

Question 4.4: Results (5 Points)

Set $K = 10$, $\alpha = 1$, $\beta_1 = 0.01$, and $\beta_2 = 0.01$. Run your implemented Gibbs sampler to collect a sample of $\boldsymbol{z}^b$ and $\boldsymbol{z}^h$. Use it to estimate the $\boldsymbol{\pi}$'s, $\boldsymbol{\theta}^b$'s and $\boldsymbol{\theta}^h$'s. Show the top-10 words with the highest probabilities in each $\boldsymbol{\theta}^b_k$ and each $\boldsymbol{\theta}^h_k$. Discuss whether these topics are meaningful.

Note: You can use a single sample of $\boldsymbol{z}$ to estimate the $\boldsymbol{\pi}$'s, $\boldsymbol{\theta}^b$'s and $\boldsymbol{\theta}^h$'s, although theoretically we should collect a set of samples. To obtain a good sample of $\boldsymbol{z}$, you need to run Gibbs sampling for a number of iterations first (to reach the stationary/invariant state).