CSE8803/CX4803 - Homework 4 - Spring 2022

In this homework, you will have one theory assignment and two coding assignments.

- Please submit the `pdf` file that includes your answer of the theory question on Canvas.
- Please submit the `hw4_1.py` and `hw4_2.ipynb` files on Gradescope (https://www.gradescope.com/courses/376874).

1 Neural network [10 pts]

You have seen in our lectures that the non-linearity and multi-layer structure of a neural network increase the capacity to model complex data. For example, a neural network is able to classify points which cannot be perfectly classified by a *linear* classifier. In this problem, you will need to design a neural network that is more "powerful" than a linear classifier.

x_1	x_2	y
1	1	-1
1	0	+1
0	1	+1
0	0	-1

Table 1: Input data

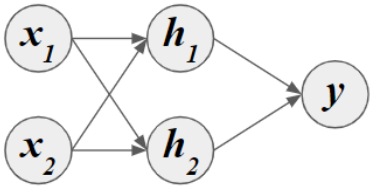

Figure 1: One-layer neural network

Suppose you a given four points on a 2D plane, $\{x_1^{(i)}, x_2^{(i)}\}_{i=1}^4$. Each point is associated with a binary label $y^{(i)} \in \{-1, +1\}$, as shown in Table 1. First plot these points and convince yourself that there is no linear classifier (linear line) that can perfectly classify the four points. Now we want to show that a simple neural network, even with only one layer of hidden units, can correctly classify those points. Consider the neural network in Figure 1, which has 2 inputs (x_1 and x_2), 2 hidden nodes (h_1 and h_2), and one output. The network can be summarized by the following functions.

$$\begin{aligned} z_1 = h_1(x_1, x_2) &= \texttt{sgn}(W_{11}x_1 + W_{12}x_2 + b_1) \\ z_2 = h_2(x_1, x_2) &= \texttt{sgn}(W_{21}x_1 + W_{22}x_2 + b_2) \\ o &= \texttt{sgn}(W_{31}z_1 + W_{32}z_2 + b_3), \end{aligned}$$

where W_{k1}, W_{k2}, and b_k ($k = 1, 2, 3$) are the weights and biases of the neural network. Here, we use the sign function `sgn` as the activation function (`sgn`(x) returns +1 if x is positive, and -1 otherwise). Please find a setting of all parameters (W_{k1}, W_{k2}, and b_k, $k = 1, 2, 3$) for this neural network that will perfect classify the four points.

Answer: Key observation: if we write the y values in $\{0, 1\}$ instead of $\{-1, +1\}$ (it does not matter whether we write y in $\{0, 1\}$ or $\{-1, +1\}$, because we can always use the bias term b_k to offset it to our desired value), our goal is to build a neural network f such that $f(1, 1) = 0, f(1, 0) = 1, f(0, 1) = 1, f(0, 0) = 0$. Note that f is exactly the XOR function! To compute $y = x_1$ XOR x_2, we can first compute $z_1 = x_1$ OR x_2 and $z_2 = \text{NOT}(x_1$ AND $x_2)$, then compute $y = z_1$ AND z_2. For example, $z_1 = x_1$ OR x_2 can be computed by $z_1 = \text{sgn}(x_1 + x_2 - 0.5)$ (you can verify that $z_1 = 1$ if and only if either $x_1 = 1$ or $x_2 = 1$). You can

find the parameters for the AND function following a similar idea. The feasible values of parameters are not unique. One possible answer is:

$$z_1 = \text{sgn}(x_1 + x_2 - 0.5)$$
$$z_2 = \text{sgn}(x_1 + x_2 - 1.5)$$
$$o = \text{sgn}(3z_1 + 2z_2 - 2.5)$$

(Rubric: Different solutions are possible. Any parameters that perfectly classify the four points receive full points.)

2 Protein structure and sequence co-evolution [15 pts]

In our lecture, we have seen that sequence co-evolution relationships can be used to assist the protein structure prediction. Given a protein sequence $\mathbf{x} = (x_1, x_2, \ldots, x_L)$ for which we want to predict the structure, we can search its homologous (evolutionarily related) sequences in the database and form a multiple sequence alignment (MSA). Next, we can use a generative model called Markov Random Field (MRF, also known as Potts model, Ising model, or undirected graphical model) to calculate the probability distribution for sequences in the MSA. As introduced in the lecture, the probability of a sequence x of length L is defined as

$$p(\mathbf{x}) = \frac{1}{Z} \exp[E(\mathbf{x})], \tag{1}$$

where Z is a normalization constant that ensures $p(\mathbf{x})$ is a probability in $[0, 1]$. In MRF, $E(\mathbf{x})$ is defined using two groups of parameters as following

$$E(\mathbf{x}) = \sum_i e_i(x_i) + \sum_{i<j} e_{ij}(x_i, x_j), \tag{2}$$

where $x_i \in \Sigma$ is the amino acid of sequence $\mathbf{x}$ at the i-th position and Σ is the alphabet of MSA (e.g., 20 possible amino acids and the gap). The parameters $e_i(x_i)$ are called single-site potentials and parameters $e_{ij}(x_i, x_j)$ are called pairwise potentials[1]. Now let us understand this model in more detail by considering the following questions.

(1) [2pts] To make $p(\mathbf{x})$ a valid probability distribution, the normalization constant Z can take the form $Z = \Sigma_{x \in \mathcal{S}} \exp[E(\mathbf{x})]$ for some set of sequences $\mathcal{S}$. What sequences should be included in $\mathcal{S}$ such that $p(\mathbf{x})$ gives the probability of a sequence x of length L? How large is $|\mathcal{S}|$?

(2) [3pts] If we build an MRF model described in Eq, 2 for an MSA of sequences with length L, and assume the alphabet is $|\Sigma| = q$, how many parameters does this MRF have?

(3) [2pts] In our lecture, we mentioned that the single-site potentials e_i can reflect the preference of an amino acid appearing at position i. For example, if $e_i(x_i)$ has a larger value compared to other $e_i(x_j)$ $(j \neq i)$, then it means the amino acid x_i will have a higher likelihood showing at position i. We also mentioned that the pairwise potentials e_{ij} can reflect co-evolving residue pairs. Now, if we have built an MRF model and learned all its parameters, and suppose that residues i and j are co-evolving pairs, while residues u and v are non co-evolving pairs. Please describe, in your own words, how would the values in e_{ij} (note that e_{ij} is a matrix with size $|\Sigma| \times |\Sigma|$) be different from those in e_{uv} ? In other words, intuitively, are there any signals/patterns in e_{ij} that can help you tell which residue pairs are co-evolving?

(4) [3pts] Based on the above intuition, can you design a "co-evolution score" $c(i, j)$ using e_{ij} such that $c(i, j)$ has a higher value when residues are co-evolving than when they are not?

[1] Note that the summation of e_{ij} here is over $\{i < j\}$, which simplifies the summation over $\{i \neq j\}$, due to symmetry

(5) [5pts] Sometimes we may be interested in comparing which sequences are more plausible, or more likely to appear in the evolution history (i.e., with a relatively higher probability $p(\mathbf{x})$). Consider a sequence $\mathbf{x}$ where the k-th position is amino acid a, i.e., $\mathbf{x} = (x_1, \ldots, x_k = a, \ldots, x_L)$, and another sequence $\mathbf{x}'$ that differs with $\mathbf{x}$ by only one position: the amino acid at position k changed from $x_k = a$ to $x_k = b$, i.e., $\mathbf{x}' = (x'_1, \ldots, x'_L) = (x_1, \ldots, x_k = b, \ldots, x_L)$. Now we want to decide which sequence is more plausible in the evolution history. One way to do this is to build an MRF on the homologous sequence of $\mathbf{x}$. With the MRF, one can compute $p(\mathbf{x})$ and $p(\mathbf{x}')$ as in Eq. 1 and compare the log-odds ratio of sequence probabilities between $\mathbf{x}$ and $\mathbf{x}'$:

$$\Delta E(\mathbf{x}', \mathbf{x}) = \log \frac{p(\mathbf{x}')}{p(\mathbf{x})} = E(\mathbf{x}') - E(\mathbf{x}) \tag{3}$$

If $\Delta E(\mathbf{x}', \mathbf{x}) > 0$, or equivalently $p(\mathbf{x}') > p(\mathbf{x})$, that means $\mathbf{x}'$ has a higher likelihood of occurrence than $\mathbf{x}$ in the evolution. Now you want to compute the value of $\Delta E(\mathbf{x}', \mathbf{x})$ based on Eqs. 1, 2, and 3. Suppose the MRF model has been built for that MSA data, and all its parameters (e_i and e_{ij}) are stored in an oracle that you can query. Each time, you can query the oracle for the value of a *single* parameter $e_i(a)$ or $e_{ij}(a, b)$ for any position i and any amino acid types a or b. You goal is to minimize the number of queries you make. What is the minimum number of queries you have to make to compute $\Delta E(\mathbf{x}', \mathbf{x})$? And what are those parameters you need to query? You need to justify you answer.

Answer:

(1) $\mathcal{S}$ should be the set of all posssible sequences with length L. $|\mathcal{S}| = |\Sigma|^L$.
(Rubric: $|\mathcal{S}| = 20^L$ can also receive full score.)

(2) For e_i, we have q parameters at each of the L positions. For e_{ij} we have $L(L-1)/2$ unique pairs for different positions i and j, and for each pair we have q^2 parameters. Therefore, the total number is $L \cdot q + L(L-1)/2 \cdot q^2$.

(3) If residues i and j are co-evolving, there are must be one or multiple amino acid pairs $\{(a, b)\}_{k=1}^m$ that are co-occuring in the MSA at those two positions. Therefore, $e_{ij}(a_k, b_k)$ would have a value larger than expected. In contrast, if residues i and j are non co-evolving (or independent), in e_{ij} there should not be an enrichment for some amino acid pair, and almost all values in e_{ij} should be very close to the expected value (say, the average e_{ij} value over all pairs of positions and all pairs of amino acid types).

(4) The are many ways to design such a score. The core idea is to design some score that would be large if e_{ij} contains large values. One example is to compute the L_2-norm of e_{ij}, i.e.,

$$w(i, j) = \sqrt{\sum_{a \in \Sigma'} \sum_{b \in \Sigma'} e_{ij}(a, b)^2},$$

where Σ' is the set of all 20 amino acids (i.e., excluding the gap). To correct for biases at individual positions, we then subtract the row, column, and full matrix average of $w(i, j)$ to get the final co-evolution score:

$$c(i, j) = w(i, j) - \frac{\bar{w}(i, \cdot)\bar{w}(\cdot, j)}{\bar{w}(\cdot, \cdot)},$$

where $\bar{w}(i, \cdot)$, $\bar{w}(\cdot, j)$, and $\bar{w}(\cdot, \cdot)$ are the row, column, and full matrix average of $w(i, j)$, respectively. This correction is also known as "average product correction".

(Rubric: Different solutions are possible. Solutions without bias correction, e.g., only $w(i, j)$, can still receive full score.)

(5) Key observations: Compare the definitions of $E(\mathbf{x}')$ and $E(\mathbf{x})$ and we may notice that for both e_i and e_{ij}, only the terms related to the k-th position changed their values, and other terms remained

unchanged. By definition, we have

$$\begin{aligned}
\Delta E(\mathbf{x}', \mathbf{x}) &= E(\mathbf{x}') - E(\mathbf{x}) \\
&= \left[\sum_i e_i(x_i) + \sum_{i<j} e_{ij}(x_i, x_j)\right] - \left[\sum_i e_i(x_i') + \sum_{i<j} e_{ij}(x_i', x_j')\right] \\
&= \left[\sum_i e_i(x_i) - \sum_i e_i(x_i')\right] + \left[\sum_{i<j} e_{ij}(x_i, x_j) - \sum_{i<j} e_{ij}(x_i', x_j')\right] \\
&= [e_k(a) - e_k(b)] + \left[\left(\sum_{i<k} e_{ik}(x_i, a) + \sum_{i>k} e_{ki}(a, x_i)\right) - \left(\sum_{i<k} e_{ik}(x_i, b) + \sum_{i>k} e_{ki}(b, x_i)\right)\right]
\end{aligned}$$

Therefore, we only need to query

- $e_i(a)$ and $e_i(b)$. (2 terms)
- $e_{ik}(x_i, a)$ for $i < k$; and $e_{ki}(a, x_i)$ for $i > k$. $(L - 1$ terms)
- $e_{ik}(x_i, b)$ for $i < k$; and $e_{ki}(b, x_i)$ for $i > k$. $(L - 1$ terms)

In total, $2 + (L - 1) + (L - 1) = 2L$ queries.

3 Coding assignmnet: Nussinov Algorithm [10 pts]

In this question, you will try to implement Nussinov algorithm to infer RNA secondary structure. Please read the description in `hw4_1.py` carefully and finish the code. Notes:

- Before submitting to Gradescope, you can test your algorithm on the RNA sequences in `test_data` (corresponding code already written for you). The expected output is (newlines stripped; there can be multiple solutions, you only need to output one of them):
  - `result_0.txt`: (0, 7) (1, 6)
  - `result_1.txt`: (1, 6)
  - `result_2.txt`: (1, 14) (2, 12) (3, 11) (4, 9)
- Please don't add extra imports or change the input/output part. However, you can add new utility functions for tracing back.

4 Coding assignment: Variational Autoencoder [15 pts]

In this question, you will be implementing variational autoencoder from scratch. Please follow the instruction of `hw4_2.ipynb` and finish the code. You will again be using human peripheral blood mononuclear cells (PBMCs) dataset [1]. Again, we recommend you to use `Google colab` for this question. Notes:

- There are some additional questions at the end of the notebook, please write down your answer of those questions in the pdf file that you submit.
- Please follow the instructions in the notebook. Only change the code in sections marked by `COMPLETE THE METHOD`.
- You can change the learning rate, the `beta` hyperparameter, and the shape of the middle layer. You may need to be patient when training a VAE.

References

[1] Hyun Min Kang, Meena Subramaniam, Sasha Targ, Michelle Nguyen, Lenka Maliskova, Elizabeth McCarthy, Eunice Wan, Simon Wong, Lauren Byrnes, Cristina M Lanata, Rachel E Gate, Sara Mostafavi, Alexander Marson, Noah Zaitlen, Lindsey A Criswell, and Chun Jimmie Ye. Multiplexed droplet single-cell RNA-sequencing using natural genetic variation. *Nat. Biotechnol.*, 36(1):89–94, January 2018.