CSCI567 2017 Homework Assignment 5

Due date and time Nov. 19th (Sunday), 11:59pm

Starting point Your repository will have a directory 'homework5/'. Please do not change the name of this repository or the names of any files/folders we have added to it. Please perform a `git pull` to retrieve these files. You will find within 'homework5/':

- Two python scripts `hmm.py` and `tsne.py`, which you will be amending by adding code for questions in Sect. 3 and Sect. 4, respectively.
- Three scripts `hmm.sh`, `pca.sh`, `tsne.sh`. You will use them to generate output files.
- Datasets: `hmm_model.json`, `mnist2500_X.txt`, and `mnist2500_labels.txt`.

Submission instructions The following will constitute your submission (in 'homework5/'):

- A PDF report named `firstname_lastname_USCID.pdf`, which contains your solutions for questions in Sect. 1. For your written report, your answers must be typeset with LaTeX and generated as a PDF file. No handwritten submission will be permitted. **Please also write your full name and USC ID in the pdf file.**
- The python scripts `hmm.py` and `tsne.py`, amended with the code you added for Sect. 3 and Sect. 4, respectively. Be sure to commit your changes!
- One .txt file, `hmm.txt` which will be the output of `hmm.sh`.
- Two .npy files, `pca.npy` and `Y.npy`, which will be the outputs of `pca.sh` and `tsne.sh`, respectively.

Note that if your submission does not include the .txt and .npy files mentioned above, we will assume that you did not do the corresponding parts of the homework.

Collaboration You may discuss with your classmates. However, you need to write your own solutions and submit separately. Also in your written report, you need to list with whom you have discussed for each problem. Please consult the syllabus for what is and is not acceptable collaboration.

Algorithmic component

1 Hidden Markov Models

[Recommended maximum time spent: 2 hours]

In this problem, we want to fit DNA sequence data with a generative model. In particular, we assume that they are generated by a hidden Markov model (HMM). Let $\mathbf{X}_{1:N} = [X_1 X_2 \dots X_N]$ be random variables corresponding to a DNA sequence of length N, controlled by hidden states $\mathbf{Z}_{1:N} = [Z_1 Z_2 \dots Z_N]$. Each X_n takes a value in $\{A, C, G, T\}$ and each Z_n takes one of the two possible states $\{s_1, s_2\}$. This HMM has the following parameters $\mathbf{\Theta} = \{\pi_i, a_{ij}, b_{ik}\}$ for $i \in \{1, 2\}$, $j \in \{1, 2\}$, and $k \in \{A, C, G, T\}$:

- Initial state distribution π_i for $i = 1, 2$:

$$\pi_1 = P(Z_1 = s_1) = 0.7; \quad \pi_2 = P(Z_1 = s_2) = 0.3. \tag{1}$$

- Transition probabilities $a_{ij} = P(Z_{n+1} = s_j | Z_n = s_i)$ for any $n \in \mathbb{N}^+, i = 1, 2$ and $j = 1, 2$:

$$a_{11} = 0.8, \quad a_{12} = 0.2, \quad a_{21} = 0.4, \quad a_{22} = 0.6. \tag{2}$$

- Emission probabilities $b_{ik} = P(X_n = k | Z_n = s_i)$ for any $n \in \mathbb{N}^+, i = 1, 2$ and $k \in \{A, C, G, T\}$:

$$b_{1A} = 0.4, \quad b_{1C} = 0.1, \quad b_{1G} = 0.4, \quad b_{1T} = 0.1; \tag{3}$$

$$b_{2A} = 0.2, \quad b_{2C} = 0.3, \quad b_{2G} = 0.2, \quad b_{2T} = 0.3. \tag{4}$$

We observe a sequence $\mathbf{O}_{1:6} = [o_1 o_2 \dots o_6] = [AGCGTA]$, please answer the following questions with step-by-step computations.

Q1.1 Probability of an observed sequence Compute $P(\mathbf{X}_{1:6} = \mathbf{O}_{1:6}; \mathbf{\Theta})$.

Q1.2 Most likely explanation Compute $\boldsymbol{z}^*_{1:6} = [z^*_1 z^*_2 \dots z^*_6] = \arg\max_{\boldsymbol{z}_{1:6}} P(\boldsymbol{Z}_{1:6} = \boldsymbol{z}_{1:6} | \mathbf{X}_{1:6} = \mathbf{O}_{1:6}; \mathbf{\Theta})$.

Q1.3 Prediction Compute $x^* = \arg\max_x P(X_7 = x | \mathbf{X}_{1:6} = \mathbf{O}_{1:6}; \mathbf{\Theta})$.

What to submit: Mathematical derivations and numerical results for each of the problems above. You should derive them manually.

Reminder: Do not have any spaces in your filename. If you are considering putting a space, put an underscore instead. Do not have more than one PDF file in your homework5 directory.

Programming component

2 High-level descriptions

2.1 Datasets

In Sect. 3, we will play with a small hidden Markov model. The parameters of the model are given in `hmm_model.json`. In Sect. 4, you will perform dimensionality reduction and visualize a subset of **MNIST**: 784-dimensional feature vectors in `mnist2500_X.txt` and their corresponding labels in `mnist2500_labels.txt`.

2.2 Tasks

You will be asked to implement hidden Markov model (HMM) inference procedures and the t-distributed stochastic neighbor embedding (t-SNE) algorithm. Specifically, you will

- For Sect. 3: Finish implementing the function `forward`, `backward`, `seqprob_forward`, `seqprob_backward`, and `viterbi`. Refer to `hmm.py` for more information.
- For Sect. 4: Finish implementing the function `pca`, `compute_Q` and `compute_gradient`. Refer to `tsne.py` for more information.
- Run the scripts `hmm.sh`, `pca.sh`, and `tsne.sh` after you finish your implementation. `hmm.sh` will output `hmm.txt`. `pca.sh` will output `pca.npy`. `tsne.sh` will show visualization of the dataset and will also output `Y.npy`.
- Add, commit, and push `hmm.py` and `tsne.py`, and the files that you have generated.

As in the previous homework, you are not responsible for loading/pre-processing data.

2.3 Cautions

Please do not import packages that are not listed in the provided code. Follow the instructions in each section strictly to code up your solutions. **Do not change the output format**. **Do not modify the code unless we instruct you to do so**. A homework solution that does not match the provided setup, such as format, name, initializations, etc., **will not** be graded. It is your responsibility to **make sure that your code runs with the provided commands and scripts on the VM**. Finally, make sure that you **git add, commit, and push all the required files**, including your code and generated output files.

3 Hidden Markov Models

In this problem, you are given parameters of a small hidden Markov model and you will implement three inference procedures, similar to what you have done manually in the first question. In `hmm_model.json`, you will find the following model parameters:

- π: the initial probabilities, $\pi_i = P(Z_1 = s_i)$;
- A: the transition probabilities, with $a_{ij} = P(Z_t = s_j | Z_{t-1} = s_i)$;

- B: the observation probabilities, with $b_{ik} = P(X_t = o_k | Z_t = s_i)$.

Now we observe a sequence O of length L. Your task is to write code to compute probabilities and infer about the possible hidden states given this observation. In particular, first in **Q3.1** and **Q3.2**, we want to compute $P(x_1, \ldots, x_L = O)$, the probability of observing the sequence. You should use the forward algorithm and the backward algorithm to achieve that. Then in **Q3.3**, we infer the most likely state path. Note that your code might be tested against different parameter sets/observation sequences at grading time.

Q3.1 Please finish the implementation of the functions `forward()` and `backward()` in `hmm.py`:

- `forward(`π, A, B, O`)` takes the model parameters π, A, B and the observation sequence O as input and output a numpy array α, where $\alpha[j,t] = \alpha_t(j) = P(Z_t = s_j, x_{1:t})$.
- `backward(`π, A, B, O`)` takes the model parameters π, A, B and the observation sequence O as input and output a numpy array β, where $\beta[j,t] = \beta_t(j) = P(Z_t = s_j, x_{t+1:T})$.

Please follow the slides 35, 36 in `lec18.pdf` for your implementation.

Q3.2 Now we can calculate $P(x_1, \ldots, x_L = O)$ from the output of your forward and backward algorithms. Please finish the implementation of the function `seqprob_forward()` and `seqprob_backward()` in `hmm.py`. Both of them should return $P(x_1, \ldots, x_L = O)$.

Q3.3 We want to compute the most likely state path that corresponds to the observation O. Namely,

$$\boldsymbol{k}^* = (k_1^*, k_2^*, \cdots, k_L^*) = \arg\max_{\boldsymbol{k}} P(s_{k_1}, s_{k_2}, \cdots, s_{k_L} | x_1, x_2, \cdots, x_L = O).$$

Please implement the Viterbi algorithm in `viterbi()` in `hmm.py`. The function `viterbi(`π, A, B, O`)` takes the model parameters π, A, B and the observation sequence O as input and output a list `path` which contains the most likely state path $\boldsymbol{k}^*$ (in terms of the state index) you have found.

What to do and submit: After you finish each task in Q3.1/3.2 and Q3.3 in `hmm.py`, run the script `hmm.sh`. It will generate `hmm.txt`. Add, commit, and push both `hmm.py` and `hmm.txt` before the due date.

4 Dimensionality Reduction

In this question, you will implement t-Distributed Stochastic Neighbor Embedding (t-SNE) [1], a (prize-winning) technique for dimensionality reduction that is particularly well suited for the visualization of high-dimensional datasets. t-SNE maps high-dimensional data points $x_1, x_2, \ldots, x_N$ into two or three-dimensional *embeddings* $y_1, y_2, \ldots, y_N$ that can be displayed in a scatter plot. Unlike PCA that you learned in class, t-SNE is a *non-linear* dimensionality reduction technique. It is widely used (more than 3,300 citations and counting). If you are curious about how to use it effectively, check this out https://distill.pub/2016/misread-tsne/.

Intuitively, we want the low-dimensional data points to reflect similarities of their corresponding data points in the high-dimensional space. In other words, after applying the mapping, similar data points are still near each other and dissimilar data points are still far apart. In t-SNE, this is achieved in two steps. First, t-SNE constructs a joint probability distribution P over pairs of high-dimensional data points in such a way that similar points have a high probability of being picked, whilst dissimilar points have an extremely small probability. Second, t-SNE similarly defines a joint probability distribution Q over pairs of low-dimensional data points and then minimizes the Kullback-Leibler (KL) divergence between P and Q. You could think of KL-divergence as a measure of how one probability distribution diverges from another.

Formally, let $p_{ij} = P(x_i, x_j)$ and $q_{ij} = Q(x_i, x_j)$. Then, we minimize

$$C = KL(P||Q) = \sum_i \sum_j p_{ij} \log \frac{p_{ij}}{q_{ij}}, \tag{5}$$

where we define each of the above terms below. Please refer to the original paper [1] for justifications of their choice of probability distributions as well as their choice of objective.

We define the pairwise similarities p_{ij} between x_i and x_j in the high-dimensional space as

$$p_{ij} = \frac{p_{j|i} + p_{i|j}}{2N}, \text{where } p_{j|i} = \frac{\exp(-\frac{\|x_i - x_j\|_2^2}{2\sigma_i^2})}{\sum_{k \neq i} \exp(-\frac{\|x_i - x_k\|_2^2}{2\sigma_i^2})}. \tag{6}$$

where σ_i^2 is the variance of a Gaussian distribution that is centered at x_i. The conditional probability $p_{j|i}$ could be interpreted as the probability of x_i *picking* x_j as its neighbor.

Moreover, we define the pairwise similarities q_{ij} between y_i and y_j in the low-dimensional space with a Student t-distribution with one degree of freedom:

$$q_{ij} = \frac{(1 + \|y_i - y_j\|_2^2)^{-1}}{\sum_{k \neq l}(1 + \|y_k - y_l\|_2^2)^{-1}}. \tag{7}$$

Finally, we set both p_{ii} and q_{ii} to zeros, as we are only interested in pairwise similarities.

We use gradient descent to minimize the cost function in Eq. (5). The gradient is given by

$$\frac{\partial C}{\partial y_i} = 4 \sum_j (p_{ij} - q_{ij})(y_i - y_j)(1 + \|y_i - y_j\|_2^2)^{-1}. \tag{8}$$

(Can you verify that this is indeed the formula for the gradient?)

Q4.1 We first apply PCA to our very high-dimensional data points. This in practice helps speed up the computation and reduce noise in the data points before we apply t-SNE. Finish the implementation of PCA in function `pca` in `tsne.py` to map the data points of dimensionality 784 to 50.

What to do and submit: Finish the implementation of function `pca`. Run the script `pca.sh`. It will output a file `pca.npy`. Add, commit and push the `pca.npy` and your modified `tsne.py` before the due date.

Q4.2 After PCA, finish the implementation of the t-SNE algorithm. In `tsne.py`, you are given the code to compute p_{ij}, and you need to finish the code that computes q_{ij} in function `compute_Q`. You may need to refer to Eq. (7).

What to do and submit: Finish the implementation of function `compute_Q`. Add, commit, and push the modified your modified `tsne.py` before due date.

Q4.3 Compute the gradient $\frac{\partial C}{\partial y_i}$ using Eq. (8) in function `compute_gradient` in `tsne.py`. Finally, running the script `tsne.sh` will show cool t-SNE visualization the data in the 2-dimensional space.

What to do and submit: Finish the implementation of function `compute_gradient`. Run the script `tsne.sh`. It will output a file `Y.npy`. Add, commit and push `Y.npy` and your modified `tsne.py` before due date.

Note: Feel free look at the Matlab implementation of t-SNE at `https://lvdmaaten.github.io/tsne/code/tSNE_matlab.zip`, but you are not permitted to look at any other online implementation of t-SNE, here and elsewhere.

References

[1] L. van der Maaten and G. Hinton. Visualizing data using t-sne. *Journal of Machine Learning Research*, 9(Nov):2579–2605, 2008.