CSE, IITB

CS 335 - AI & ML
Lab 6 - Graphical Models & Search (9 marks)

Please read the following important instructions before getting started on the assignment

1. The assignment should be completed individually. Do not look at solutions to this assignment or related ones on the Internet.
2. Solutions to the theory questions must be submitted in a single pdf file.
3. All the hyperparameters must be specified in pdf file under **Hyperparameter** section and resources consulted must be duly listed in the **References** section of the pdf file. Remember to embed plots, if any, inside pdf. **Do not upload multiple pdf files.**
4. **Upload Guidelines**: Put all the assignment related files in folder with the convention **roll_no** and .zip the folder (**no tar.***). Also provide **readme.txt** for Part 2 in which you mention all files you create. Directory Structure:
   your_roll_no/
   - roll_no.pdf
   - *.py files [as described in tasks]
   - *.txt files [as described in tasks] -readme.txt + [output].txt files
5. **Not following folder guidelines will attract penalty.**
6. All source code are checked with code plagiarism checker. Strict action will be taken against the defaulters.
7. Comment out all the print statements from the source before submission. Such mistakes will attract penalty.

Q1 : EM for Mixture of Gaussians (3 marks)

Task 1 : Implementing EM for GMM (3 marks)

Refer to the problem 6 (and if required problem 5) of Tutorial 8 for the algorithm. Complete `E_step` and `M_step` in GMM.py. You can change the initialization. Use test.py to visualize the data. Submit the plot in the pdf. Also test your code on data with different k, mu and $sigma$ before submission and write your observations

Q2 : POS Tagging using HMM (6 marks)

Understanding the data

In this task you are going to use HMMs for POS tagging (as a related motivational problem that was also discussed in the class, you could also look at Problem 2 of Tutorial 8). Ev-

ery line of training file consists of word_tag forms as you can see in `wiki-en-train.norm_pos`.

Transition probability $P(T_{j,t+1}|T_{i,t})$ is the probability of predicting T_j given that the tag at previous time step t is T_i. The number of tags are finite, say N. So the transition matrix(of size N x N) is defined in which the element (i,j) consists of the transition probability from tag T_i to T_j.

The transition probabilities can be calculating using

$$P(T_j|T_i) = \frac{c(T_j, T_i)}{\sum_{k=1}^{N} c(T_k, T_i)}$$

where the T_i is the tag number i and $c(T_j, T_i)$ is the count of instance token (in a sentence) having tag T_j given the previous token is tagged T_i.

Emission Probabilities:

$$P(w_k|T_i) = \frac{\sum_{j=1}^{M_k} \mathbb{1}[T(w_k) = T_i]}{\sum_d \mathbb{1}[T = T_i]}$$

Out of the all the occurrences of tag T_i in the training set d , how many times the word(or token) w_k is tagged has a tag T_i. $T(w_k)$ is tag allotted to the word k in the vocabulary. Note that these probabilities are to be calculated from the training set itself.

- You need to use smoothing techniques to deal with unknown contexts in testing data. A simple linear interpolation would be enough.

  $$P_{smooth}(w_k|T_i) = (1 - \lambda) * P_{old}(w_k|T_i) + \frac{\lambda}{|V|}$$

  where $|V|$ is vocabulary size. Take $|V| = 1e + 8$. Tune λ accordingly. Take $P_{old} = 0$ if $T_i \rightarrow w_k$ is not seen.

- use token $< s >$ *and* $< /s >$ to denote start and end of the sentence respectively.

Calculate the probabilities by writing a file `train.py`
which outputs the emission and the transition probabilities in `hmm_pos.txt` in a format of your choice and can be run using the following command.
`python3 train.py --train-file wiki-en-train.norm_pos --model hmm_pos.txt`

Task 2: Inference strategies for HMMs: (6 marks)

First find out the transition[POS → POS] and emission[POS → Word] probabilities. After finding out the transition and emission probabilities you are going to use Max-Product, A* search and Beam Search for finding out the best path.

- Max-Product in HMMs(Viterbi Algorithm) - `max_Product.py`

- A* Search - `astar_Search.py`
- Beam Search - `beam_Search.py`

The command to test your code is
`python3 [your py file] --model hmm_pos.txt --input test.txt --output [output file]`
To find out the accuracy of your model use the following command:
`perl gradepos.pl ground_truth.txt [output file]`
`[your py file]` is the implementation of the inference algorithms, which initializes the HMM's using the transition and emission probabilities generated from the training set. `[Output file]`, the output files generated from consists of the tags predicted in on a per sentence basis(in test.txt) and outputs in the format required in `ground_truth.txt`. Please name the output file in the following manner:

- `infer_max.txt`
- `infer_astar.txt`
- `infer_beam.txt`

for the corresponding algorithm.

- You are expected to provide **elaborate comments** along with your code (for the 3 files). Your marks depend on whether the TAs are able to understand your code and establish its correctness.
- The test data given to you is only 50% of original test data. We are going to test your code on a different test set.
- Ensure that your code produces output in correct format such that it can be tested using the given perl script. We will be executing your code on the different test set using the same perl script.