CS6375 Project 4 Part-2 Report
sxj220028

Subpart 1: Use the Chow-Liu algorithm to learn the structure and parameters of the Bayesian network.
- I used the Chow-Liu Tree class already provided to learn all the datasets and used that tree to report the test-set log-likelihood of all datasets. The code for subpart1 is in the file subpart_1.py. The output of this subpart is present in subpart1_output.txt. The test set log-likelihood of all the datasets using the default Chow-Liu algorithm is the following:

Dataset	**Log-likehood**
accidents.test.data	-33.188
baudio.test.data	-44.375
bnetflix.test.data	-60.25
jester.test.data	-58.226
kdd.test.data	-2.295
msnbc.test.data	-6.54
nltcs.test.data	-6.759
plants.test.data	-16.524
pumsb_star.test.data	-30.807
tretail.test.data	-10.946

Subpart 2: I wasn't able to fully implement the EM algorithm with Chow-Liu trees. The incomplete code for this subpart is present in the files *subpart_2.py* and mixture_clt.py.

Subpart 3: I was able to implement the Random Forest with Chow-Liu trees as mentioned in the project document. The code for this subpart is in the files *subpart_3.py* and *clt_rf.py*. The output of five runs of this subpart is present in subpart3_output_i.txt where i is 1 to 5.
In my Random Forest implementation, I create *k* bootstrap samples from the training dataset and Chow-Liu tree instances. The *ith* Chow-Liu tree learns the *ith* bootstrap sample. As the project document suggests I used *p_i = 1/k* for the log-likelihood of the *ith* tree which is effectively just the Arithmetic mean (AM) of the all log-likelihood scores to calculate the aggregate log-likelihood score. I observed that the results from this Random-Forest implementation were only marginally better than the case when I just had one Chow-Liu tree.
Note: I tried a novel idea to calculate the aggregate log-likelihood as the Geometric mean (GM) and Harmonic mean (HM) of the individual scores, then choose the best aggregate score from the AM, GM, and HM values. Since Geometric and Harmonic means of negative values can sometimes be complicated, I instead used a slightly different way to calculate them.

```
# Default method where p_i = 1/k i.e arithmetic mean of all the individual LL
scores
am = np.mean(log_likelihoods)
# Taking Geometric mean and Harmonic mean of absolute values of all the
individual LL scores and multiply with -1 since all original values were
negative
gm = -1 * gmean(np.abs(log_likelihoods))
hm = -1 * hmean(np.abs(log_likelihoods))
return max(am, gm, hm)
```

I believe it is a reasonable method to calculate the aggregate log-likelihood score.
For hyper-parameter tuning, we have *k* from the set {20, 50, 100} and *r* from the set {5%, 10%, 20%, 30%} where *k* is the number of Chow-Liu trees in the Random Forest and *r* is the percentage of total mutual information scores which are set to zero.
As mentioned in the project document, I ran my Random Forest implementation 5 times, and the average and variance of log-likelihoods are reported below.

Dataset	**Log-likehood (Avg)**	**Log-likehood (Variance)**
accidents.test.data	-33.21	0.00003
baudio.test.data	-44.43	0.00006
bnetflix.test.data	-60.27	0.00010
jester.test.data	-58.27	0.00005
kdd.test.data	-2.3	0.000008
msnbc.test.data	-6.54	0.00011
nltcs.test.data	-6.76	0.00005
plants.test.data	-16.53	0.00003
pumsb_star.test.data	-30.82	0.00013
tretail.test.data	-10.96	0.00010

Can you rank the algorithms in terms of accuracy (measured using test set LL) based on your experiments? Comment on why you think the ranking makes sense.
- Chow-Liu Random Forest > Chow-Liu tree