Homework 4

Learning Bayesian Networks.

There are 10 datasets ordered by alphabetical order and ordered by train size.

Dataset Alphabetical Order	Dataset Size (MB)		
	Train	Valid	Test
accidents	2.7	0.368	0.553
baudio	2.86	0.39	0.585
bnetflix	2.86	0.39	0.585
jester	1.71	0.195	0.803
kdd	21.9	2.42	4.26
msnbc	9.44	1.25	1.88
nltcs	0.505	0.0674	0.101
plants	2.29	0.312	0.469
pumsb_star	3.81	0.52	0.78
tretail	5.67	0.774	1.13

Dataset Sorted by Train Size	Dataset Size (MB)		
	Train	Valid	Test
nltcs	0.505	0.0674	0.101
jester	1.71	0.195	0.803
plants	2.29	0.312	0.469
accidents	2.7	0.368	0.553
baudio	2.86	0.39	0.585
bnetflix	2.86	0.39	0.585
pumsb_star	3.81	0.52	0.78
tretail	5.67	0.774	1.13
msnbc	9.44	1.25	1.88
kdd	21.9	2.42	4.26

1. Tree Bayesian networks.

Index	Dataset	Valid Dataset Log-likelihood	Test Dataset Log-likelihood	Run Time(s)
0	accidents	-32.896648	-33.18811	5.58
1	baudio	-44.079822	-44.374902	5.718
2	bnetflix	-60.249665	-60.250346	5.9
3	jester	-58.342419	-58.226532	6.077
4	kdd	-2.526486	-2.294894	41.175
5	msnbc	-6.540025	-6.540127	19.935
6	nltcs	-6.718532	-6.759045	1.065
7	plants	-16.517598	-16.524015	4.497
8	pumsb_star	-30.92578	-30.807048	8.106
9	tretail	-10.940996	-10.946545	11.685

We implemented the Tree Bayesian Networks by using the **Chow-Liu** algorithm to learn the structure and parameters of the Network. The provided implementation of Chow-Liu tree is using the 1-Laplace smoothing to ensure that we don’t have any zeros when computing the mutual information as well as zero probabilities in the model. The implementation is very faster as it uses the **numpy.einsum (Einstein summation)** of the numpy library which provides the optimized & flexible way to compute the complex array operations.

2. Mixtures of Tree Bayesian networks using EM.

Log-Likelihood on the valid dataset using Mixture of Tree Bayesian Networks using EM

One latent variable having k values and each mixture component is a Tree Bayesian network. Thus, the distribution over the observed variables, denoted by X (variables in the data) is given by:

$$P(\mathbf{X} = \mathbf{x}) = \sum_{i=1}^{k} p_i T_i(\mathbf{X} = \mathbf{x})$$

where P_i is the probability of the ***i***-th mixture component and T_i is the distribution represented by the ***i***-th Tree Bayesian network.

Valid Dataset	K			
	2	**5**	**10**	**20**
accidents	-31.745104	-30.352457	-29.868223	**-29.404065**
baudio	-41.809687	-40.528947	-40.07817	**-39.861195**
bnetflix	-59.07827	-57.90224	-57.064702	**-56.818009**
jester	-55.434008	-53.921315	-53.425977	**-53.364487**
kdd	**-2.42186**	-2.483029	-2.466805	-2.467078
msnbc	-6.540024	**-6.535772**	-6.536489	-6.536742
nltcs	-6.718429	-6.021783	-5.965517	**-5.963121**
plants	-15.368295	-14.340616	-13.514582	**-13.279228**
pumsb_star	-27.140188	-25.337168	-24.443475	**-23.999558**
tretail	**-10.8408**	-10.902002	-10.9137	-10.9048

Average and standard deviation of the Log-Likelihood on the test dataset using Mixture of Tree Bayesian Networks using EM by running this algorithm for 5 times.

Test Dataset	**K**	**Seed1**	**Seed2**	**Seed3**	**Seed4**	**Seed5**	**Avg. LL**	**Std. LL**
accidents	20	-29.80388	-29.75698	-29.79863	-29.75997	-29.74571	-29.77303	0.02637
baudio	20	-40.09325	-40.15204	-40.06020	-40.07224	-40.12685	-40.10091	0.03816
bnetflix	20	-56.77235	-56.83449	-56.78408	-56.76971	-56.86043	-56.80421	0.04089
jester	20	-53.05623	-53.12242	-53.09731	-53.13937	-53.17389	-53.11784	0.04428
kdd	2	-2.27089	-2.25181	-2.24938	-2.27233	-2.27152	-2.26319	0.01154
msnbc	5	-6.53593	-6.53679	-6.53477	-6.53651	-6.54013	-6.53683	0.00200
nltcs	20	-6.01834	-6.01487	-6.01230	-6.01708	-6.02060	-6.01664	0.00319
plants	20	-13.16684	-13.19326	-13.11489	-13.22107	-13.09661	-13.15853	0.05226
pumsb_star	20	-23.85073	-23.81292	-23.78840	-23.80789	-23.93006	-23.83800	0.05620
tretail	2	-10.92678	-10.93016	-10.92103	-10.92981	-10.92492	-10.92654	0.00377

We used the parameters for K = {2, 5, 10, 20} and the iteration count = 50 and epsilon = 0.001 for the convergence (i.e., whichever is earlier terminates the loop). Completed the implementation of the two functions "learn(..)" and "computeLL(...)" in the code provided for the file MIXTURE CLT.py. and learns the structure and parameters of the model using the EM-algorithm (in the M-step each mixture component is learned using the Chow-Liu algorithm).

As this algorithm takes very long time to run over the all the 10 datasets as the complexity of computing the Joint Distribution probability for the Mixture is Length of dataset times the number of trees in the mixture O(d*k) so used multithreading process pool to calculate the computation of the probability which cut down by the number of processors on the system.

3. Mixtures of Tree Bayesian networks using Random Forests.

$$P(\mathbf{X} = \mathbf{x}) = \sum_{i=1}^{k} p_i T_i(\mathbf{X} = \mathbf{x})$$

Similar to the above (Item (2)) used the same technique for computing the distribution.
The best k and r values obtained from the validation set are used for testing and calculated the average and the standard deviation of the log-likelihood.

Dataset	P(i) Baseline	K	R	Avg. LL	Std. Dev LL	Run Time(s)
accidents	**Baseline**	**10**	**100**	**-33.119495**	**0.014845**	**135.016**
	Reasonable	10	100	-33.119676	0.01492	135.438
baudio	Baseline	20	150	-43.776948	0.01986	277.83
	Reasonable	**20**	**150**	**-43.776919**	**0.019859**	**276.257**
bnetflix	Baseline	20	150	-59.840093	0.008017	244.413
	Reasonable	**20**	**150**	**-59.840081**	**0.008014**	**244.124**
jester	Baseline	20	150	-57.318872	0.031609	154.56
	Reasonable	**20**	**150**	**-57.318866**	**0.031613**	**154.812**
kdd	Baseline	20	50	-2.257766	0.0012	1567.536
	Reasonable	**20**	**50**	**-2.257764**	**0.001202**	**1567.498**
msnbc	**Baseline**	**20**	**25**	**-6.527941**	**0.000979**	**795.789**
	Reasonable	20	25	-6.527948	0.000978	795.572
nltcs	**Baseline**	**20**	**50**	**-6.535583**	**0.00926**	**48.612**
	Reasonable	20	50	-6.5363	0.009359	48.903
plants	Baseline	10	150	-16.0117	0.042407	101.291
	Reasonable	**10**	**150**	**-16.01154**	**0.042498**	**101.935**
pumsb_star	**Baseline**	**20**	**100**	**-30.685462**	**0.013369**	**377.42**
	Reasonable	20	100	-30.68552	0.013379	377.746
tretail	Baseline	20	100	-10.90565	0.002138	525.537
	Reasonable	**20**	**100**	**-10.905655**	**0.002141**	**524.93**
	6 out of 10					

Developed a new function in the Chow-Liu tree CLT_class.py such that while learning we are able to set the r mutual information to zeroes and able to learn the structure and parameters of the mixture of Tree Bayesian networks using Random Forests.

Used the baseline approach of $P_i = 1 / k$ and for **the extra credit reasonable method** applied the higher weightage to the tree which has performed better on the validation dataset with very good log-likelihood scores. For those higher probability weightages to the trees applied normalization on the validation Log-likelihood scores of the trees in the mixture. And it performed better out of the 10 datasets it has performed better on the 6 datasets. For the datasets which are not very large the reasonable method performed better.

4. Mixtures of tree Bayesian networks using the gradient Boosting (Extra Credit)

A Bayesian network is a graphical model for describing a joint distribution over a set of random variables and the advantage of Bayesian networks is the ability to tune the strength of the weak learners using parameters such as number of edges and strength of prior. *Bayesian networks learns iteratively* building a prior distribution of functions over the hyperparameter space and sampling with the goal of minimizing the posterior variance of the loss surface.

Similar to the EM algorithm we initialize the random weights for the tree mixture and use a learning rate or a small coefficient. We have K Tree Bayesian Networks as the weak learners and a loss function which is the negative log-likelihood of the validation set.
Boosting algorithm sequentially finds the models $T_1, T_2, T_3, \dots T_k$ and the constants $P_1, P_2, P_3 \dots P_k$ to minimize the loss function L

$$F_i = \sum_{j \le i} Pj\ Tj(X = x)$$

Minimize the

$$L = -log(\sum_i Pi\ Ti(X = x))$$

Identify the weak learner and add it to the model with a small coefficient epsilon in each step i of the boosting algorithm. We choose the weak learner and add it to the new model's training loss as

$$\mathbf{G = F_{i-1} + \epsilon T_i}$$

$$F_{i-1} = (\sum_{j<i} Pi * Ti(X = x))$$

As per the paper as epsilon is so small in the second order term can be ignored and we are able to find the first order optimal weak learner.

Ranking based on Log-Likelihood

Index	Dataset	Test Dataset Log-likelihood	Test Dataset Avg. Log-likelihood		
		Tree Bayesian Networks	Expectation Maximization (EM)	Random Forests	
				Reasonable	Baseline
0	accidents	-33.18811	**-29.77303389**	-33.119676	-33.119495
1	baudio	-44.374902	**-40.10091435**	-43.776919	-43.776948
2	bnetflix	-60.250346	**-56.8042117**	-59.840081	-59.840093
3	jester	-58.226532	**-53.11784239**	-57.318866	-57.318872
4	kdd	-2.294894	-2.263186769	**-2.257764**	-2.257766
5	msnbc	-6.540127	-6.536826494	-6.527948	**-6.527941**
6	nltcs	-6.759045	**-6.016637091**	-6.5363	-6.535583
7	plants	-16.524015	**-13.15853409**	-16.01154	-16.0117
8	pumsb_star	-30.807048	**-23.83799831**	-30.68552	-30.685462
9	tretail	-10.946545	-10.9265408	-10.905655	**-10.90565**
Rank		**0 out of 10**	**7 out of 10**	**1 out of 10**	**2 out of 10**
		4	**1**	**3**	**2**

The higher the Log-likelihood score the better the algorithm performed. The Mixture of Tree Bayesian Networks using Expectation Maximization outperforms among all the three algorithms for datasets which are not very large (kdd, msnbc, tretail) whereas the Mixture of Tree Bayesian Networks using Random Forests outperforms for these large datasets. This Ranking may be specific to these datasets and may vary for the data which has the dependence on the features. Even though the reasonable method is overall winner in case of medium sized datasets but the baseline performed better for large ones. Also, it is evident from the table values that the Random Forests helps in reducing the standard deviation and which also means reduces the variance.