Q1 :

Dataset : Tree

Algorithm	Ch-Acc(%)	Wd-Acc(%)	LL	Time(Seconds)
Gibbs	66.21	16.67	-6.18	1
Loopy BP	67.56	16.67	-11.39	1

Dataset : treeWS

Algorithm	Ch-Acc(%)	Wd-Acc(%)	LL	Time(Seconds)
Gibbs	65.78	15.21	-6.77	2
Loopy BP	66.99	16.30	-12.59	2

Dataset : loopy

Algorithm	Ch-Acc(%)	Wd-Acc(%)	LL	Time(Seconds)
Gibbs	56.83	7.14	-7.05	2
Loopy BP	56.83	7.14	-12.03	2

Dataset : loopyWS

Algorithm	Ch-Acc(%)	Wd-Acc(%)	LL	Time(Seconds)
Gibbs	64.96	13.84	-6.51	2
Loopy BP	68.51	16.15	-13.27	2

Performance of Gibbs and Loopy BP are very similar on the problem of optical character recognition. Both algorithms are approximate inference algorithms thus they can be stopped at any point of point using some threshold. Time taken by both algorithms are far less than time taken by exact inference algorithms.

Q2 a) Results

Model : Bayesian Network

Dataset	Ava-Accuracy(%)	Avg-LL
Andes	77.18	-44.20
Hepar 2	78.10	-16.34
Insurance	80.98	-6.25

In case of bayesian network, we have modified gibbs sampler to compute the conditional probabilities efficiently, here we need to sample variables that are not given as evidence thus reducing the number of elements to be sampled and taking less time to converge on the marginal probabilities of query variables.

Q2 b) Results

Model : Markov Network

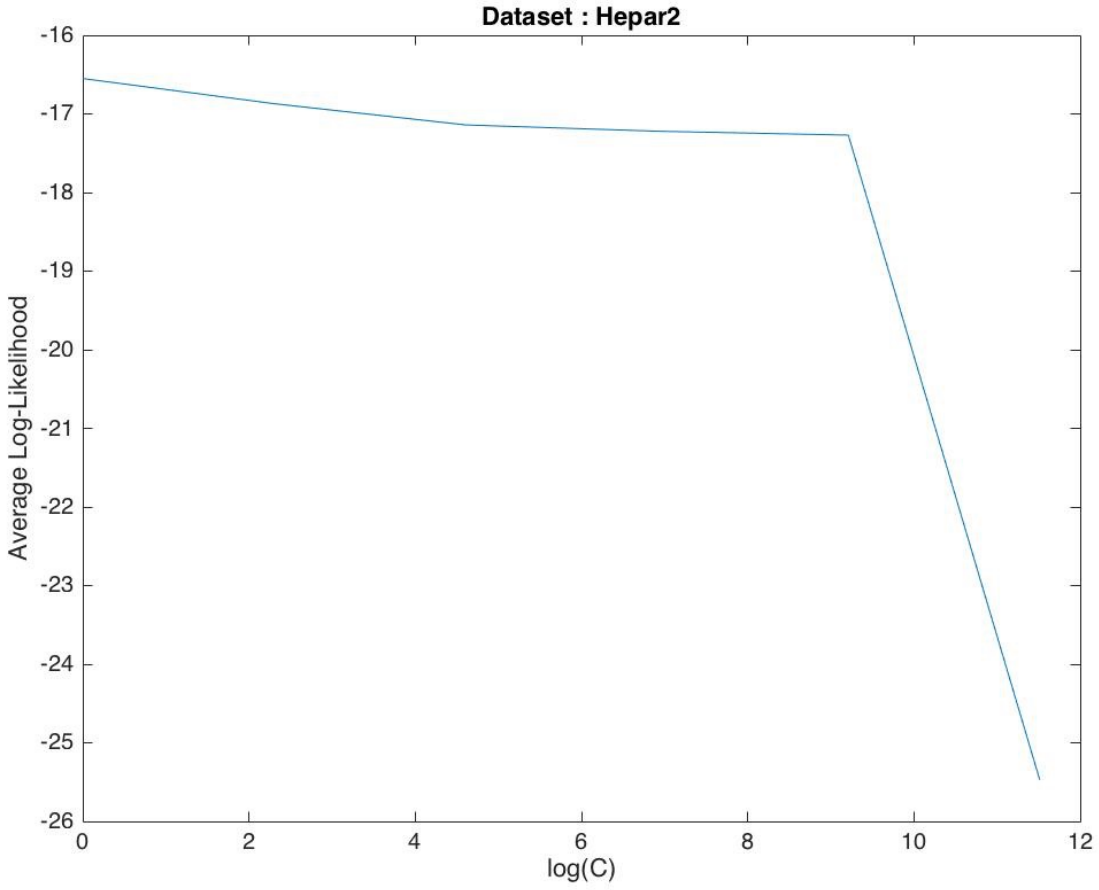
Dataset	Ava-Accuracy(%)	Avg-LL
Andes	79.10	-49.52
Hepar 2	79.88	-16.55
Insurance	76.05	-9.97

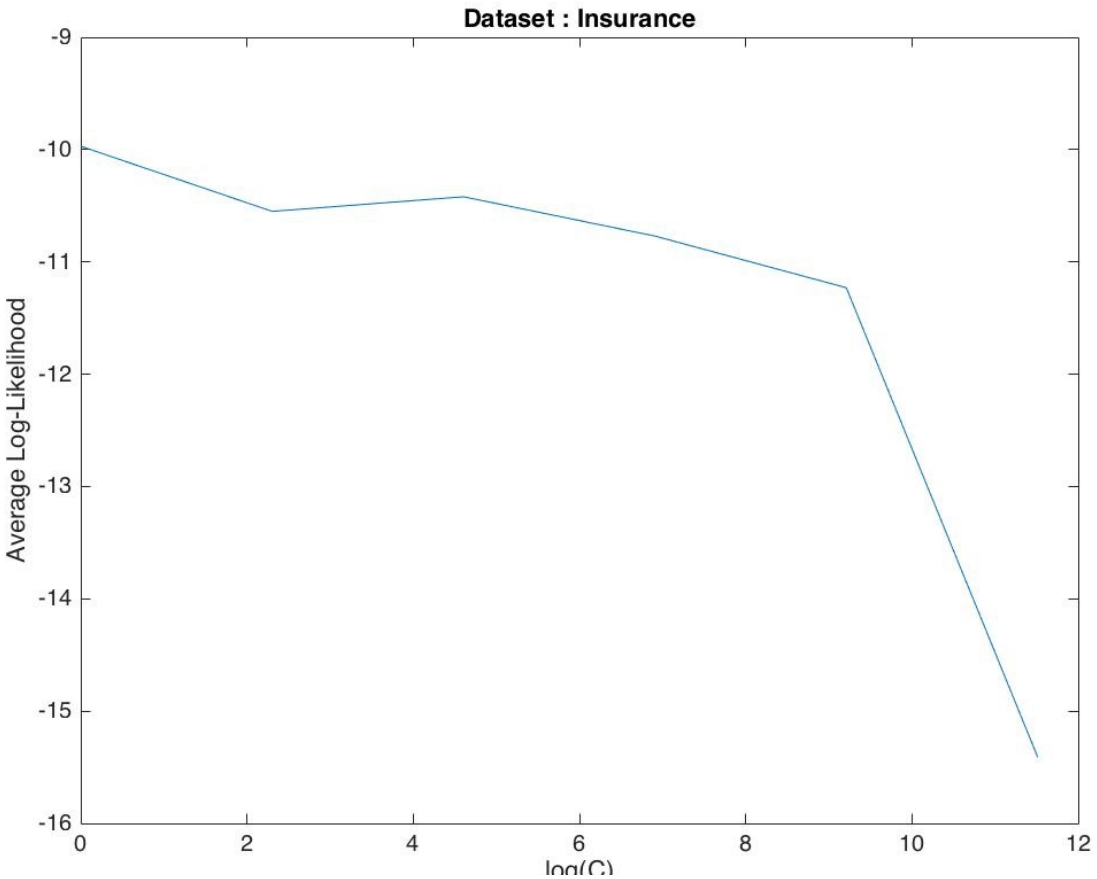
Performance of Bayesian and Markov network are very similar on the given datasets. They almost achieve similar results on both metrics. Bayesian network is more useful if their is any kind of directionality in the concerned problem which doesn't seems to be case on the given datasets.

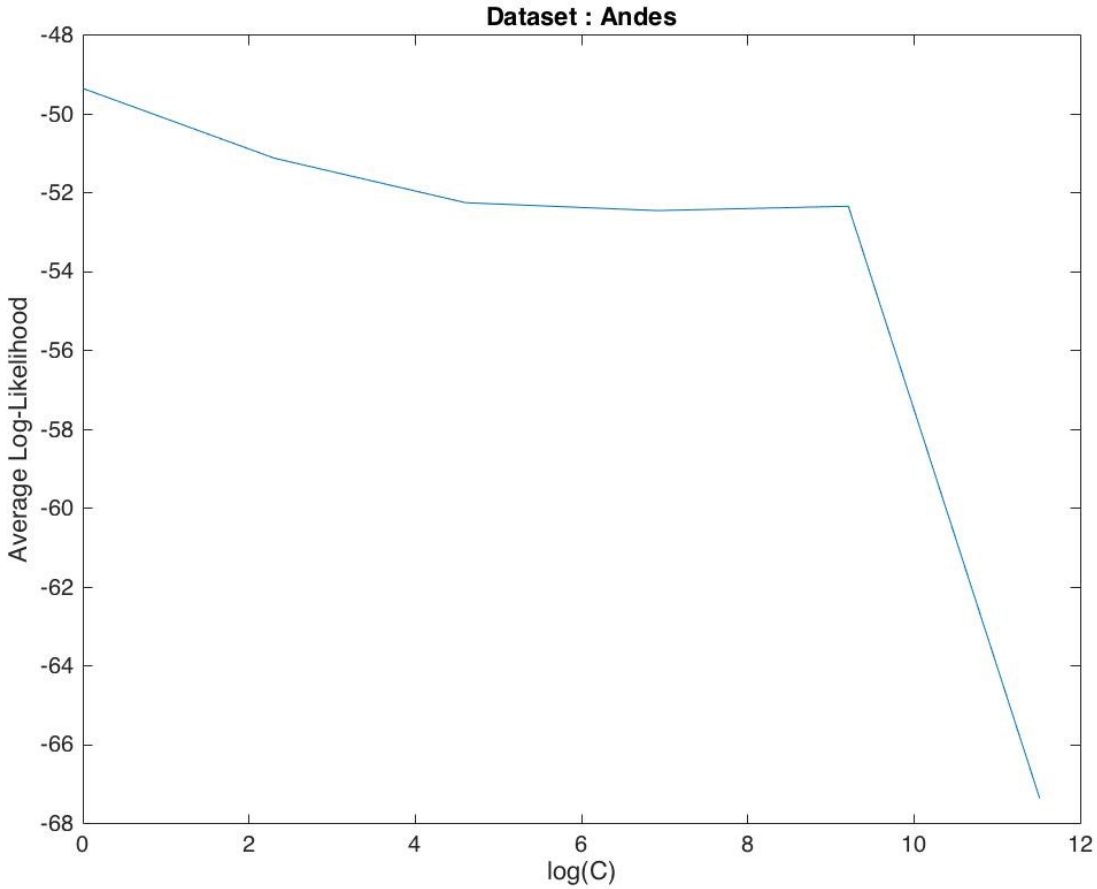
Log-likelihood with change in the value of C (weight to regularizer)

Dataset	C=1	C=10	C=100	C=1000	C=10000	C=100000
Andes	-49.52	-51.11	-52.24	-52.44	-52.33	-67.35
Hepar II	-16.55	-16.87	-17.14	-17.22	-17.27	-25.47
Insurance	-9.97	-10.55	-10.42	-10.77	-11.23	-15.41

Value of Log-Likelihood remains almost same when we vary value of C from 1 to 10^4 but when C = 10^5 which is equal to the size of dataset. The effect of regularisation is decreased as regularisation term is given by (C*lambda)/m (m = #data points as scaling factor) and it equals to the value of initial lambda thus both canceling each other (in case learning rate is 1)and not contributing towards Likelihood. As Regularisation effect decreases, the model tends to overfit and thus results in decrease in Log-Likelihood as well as accuracy. As C increases, update to the actual value of lambda decreases, as a part of lambda getting decreased by regularisation factor.




Dataset : Insurance
Average Log-Likelihood
-9
-10
-11
-12
-13
-14
-15
-16
0
2
4
6
8
10
12
log(C)


Dataset : Andes
Average Log-Likelihood
-48
-50
-52
-54
-56
-58
-60
-62
-64
-66
-68
0
2
4
6
8
10
12
log(C)