COL776 - Assignment 1

J. Shikhar Murty
2013EE10462

Implementation Language: C++

Q1.

Part-A

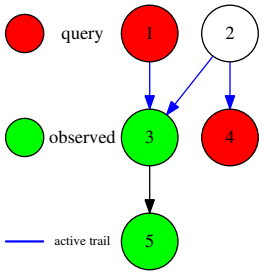
User inputs the number of nodes n and the max number of children k. The program for generating the random bayesian network, considers the nodes in the order 1...n (assuming it to be the topological order). For each node i, a number u between 1 and k is generated. If $u > (n - i - 1)$ then all the remaining nodes are considered as children of the ith node. Otherwise, the list $(i + 1...n)$ is shuffled and the first u nodes are taken. This ensures that the u children nodes are picked uniformly at random.

Part-B

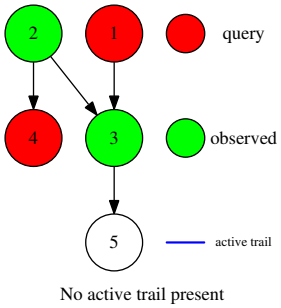
The algorithm inputs pair of nodes (X_i, X_j) and a set of observed nodes Z. I modified BFS for dseparation in the following way:

- Firstly, I perform a multi source BFS enqueueing all the nodes in Z and only following the parent pointers and store all the nodes reached in a set "valid_v". This set contains all the nodes that are potential active v-structure nodes.
- Next, I create a queue and define a state as a $(node, dir)$ pair, where **dir** is 1 if the node was reached from its parent otherwise its 0. Then I perform a BFS over an undirected version of the bayesian network, ensuring that if a node i is visited from its parent, $(i, 1)$ is pushed into the queue else $(i, 0)$ is pushed into the queue. Along with this, every time a state is dequeued for the first time, I add it to a hash table of visited states, to ensure no state is visited twice, or in other words, no node is visited more than once. This ensures the time complexity to be O(m+n). Also, I maintain a map to track back pointers for reconstructing the path. All the reachable states from the starting state $(X_i, 0)$ are added to another set, and the test of d-separation then boils down to checking if either $(X_j, 0)$ or $(X_j, 1)$ is present in the set. If so, an active trail can easily be reconstructed by following the backpointers.

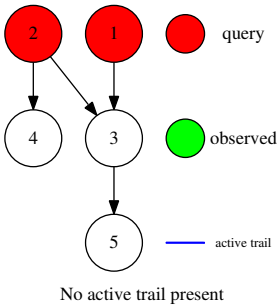
Part-C



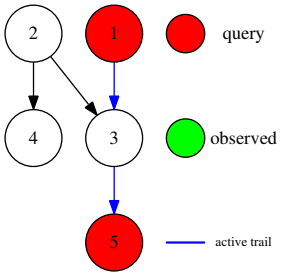
query-1



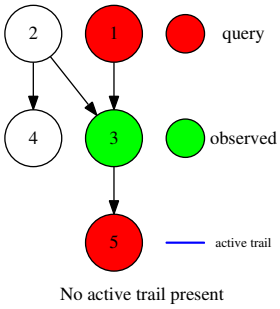
query: 2



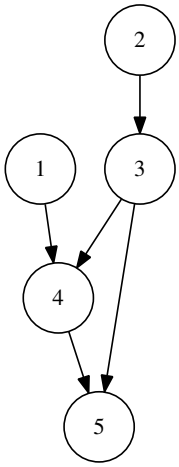
query-3



query-4



query-5



sample bn

Q2

Data structures for representing potential tables: 2D tables (vector of vectors)

Task-1

All the 3 models were implemented as separate classes with the same API in a modular way, with separate optimised implementations for each of the models. The implementation allows switching between the 3 models by taking in the model name as a command line argument.

Task-2

To perform inference, an O(n) (where n is the size of the image vector) algorithm has been implemented for both the OCR model and the Transition model. For the OCR model, inference is just picking the character that has the highest OCR potential for a given image. For the Transition model, a viterbi style algorithm was implemented with the following semantics:

$$dp[i][j] = \arg\max(w_1....w_{i-1}) \Big[P(y[1..i], x[1..i]) \Big] \text{ with } w_i = j.$$

$$dp[i+1][j] = \psi_o(img(i+1), j) * \arg\max \Big[(w_i = k) dp[i][k] * \psi_t(k, j) \Big]$$

To retrieve the word, along with the dp table, a back pointer table (bp table) was also maintained.

For the Combined model, the following algorithm was used to reduce the runtime.

- Store all indices participating in a skip link in a hash table. Let's say we have k such indices. For all 10^k assignments to these indices, run an O(n) dp based inference algorithm to figure out the best assignment to remaining characters, and the score. Possibly increment the scores based on the particular assignment to the skip link characters. (For example, if a skip link pair (i, j) is assigned the same value, increment the score by $\psi_s(i, j)$.)
- In order to calculate Z, we precompute it during inference, by using the above idea, which is to calculate the Z for all the 10^k assignments to the skip link indices. The Z obtained in this manner is then multiplied by $\psi_s(i, j)$ for every skip link pair (i, j).

Task-3

model	char accuracy	word accuracy	log-likelihood
OCR	0.539216	0.0865385	-7.80833
TRANS	0.662745	0.259615	-7.09707
COMBINED	0.711765	0.355769	-6.27953

Results on small dataset

word	OCR	TRANS	COMBINED
noon	nssn	nson	noon
toss	thhh	thad	toss
arad	arae	arad	arad
tho	thh	tho	tho
thin	thdn	thin	thin
teras	tehas	teras	teras
ratoon	raessn	rathon	ratoon
diter	eiedr	diter	diter
arad	arae	arad	arad

Example predictions

Task-4

model	char accuracy	word accuracy	log-likelihood
OCR	0.583936	0.111974	-7.87635
TRANS	0.680465	0.240402	-7.17574
COMBINED	0.708398	0.314899	-6.27186

Results on allimages1.dat

model	char accuracy	word accuracy	log-likelihood
OCR	0.57258	0.100091	-7.87404
TRANS	0.676893	0.241773	-7.17419
COMBINED	0.707208	0.318099	-6.2713

Results on allimages2.dat

model	char accuracy	word accuracy	log-likelihood
OCR	0.572488	0.0991773	-7.86572
TRANS	0.678725	0.246801	-7.1671
COMBINED	0.706292	0.31947	-6.26506

Results on allimages3.dat

model	char accuracy	word accuracy	log-likelihood
OCR	0.575785	0.114717	-7.86982
TRANS	0.68248	0.246801	-7.17084
COMBINED	0.707757	0.318556	-6.26793

Results on allimages4.dat

model	char accuracy	word accuracy	log-likelihood
OCR	0.58531	0.115631	-7.85745
TRANS	0.684587	0.26691	-7.15843
COMBINED	0.710688	0.333181	-6.25789

Results on allimages5.dat

Extra credit

Clearly, scaling the ψ_t or the ψ_o by a constant factor doesn't change the distribution and hence the performance doesn't change. Here are some of the experiments conducted and the results:

link strength	char accuracy	word accuracy
0.1	0.584314	0.153846
1.0	0.662745	0.259615
2.0	0.711765	0.355769
5.0	0.711765	0.355769
10.0	0.711765	0.355769

On changing the strength of the skip links

model	char accuracy	word accuracy	log-likelihood
TRANS	0.621569	0.221154	-6.66936
COMBINED	0.668627	0.326923	-5.86052

On squaring ψ_t (performance decreases)