COL776: Assignment #1

By

Rishit Sanmukhani (2013CS10255)

1. Conditional Independence in Bayesian Networks

a. Bayesian network is represented as ‘’Directed Acyclic Graph” in C++. Graph is stored in adjacency matrix structure. For each node, we maintain out-going edges and incoming edges.
   [Please refer q1/q1a/q1a.cpp]
   ./run.sh <number-of-nodes> <maximum-children>
b. We use the “Baseball” Algorithm (modification of Depth-First-Search) to find conditional independence.
   While doing the Depth First Search, we maintain the direction from which we have arrived at any given node. On basis of the direction (incoming/outgoing), we identify the structure in bayesian network and apply the rules of d-separation. Entire algorithm is done using single DFS. So complexity is linear (O(m+n))
   [Please refer q1/q1b/q1b.cpp]
c. All images are present in q1/q1c

2. OCR character Recognition using Graphical Models

a. Markov network is represented as “Undirected Graph” in C++. In model class, we store transition and ocr probabilities along with log probabilities. We also store the skip factor along with log factor.
   Depending upon the type of model (0, 1, 2), log prob is calculated.

b. In order to calculate Zimg, we precompute and store all the possible combination of characters for length (<=6). Later, we simply iterate over the precomputed strings and calculate log probabilities.

c. **For small data sets**:

	char-accuracy	word-accuracy	log-probability
OCR Model	53.921	8.653	-7.808
Transition Model	66.275	25.961	-7.097
Combined Model	71.176	35.576	-6.279

OCR: arena, athar, dan, netted, restes, retin, sheth, shoat, tooroo
Transition: ado, arad, ared, diter, dorter, hent, nesh, ona, orad, ortet, reader, renter, retain, rod, serai, shear, tasse, teras, thin, tho, toston, tote
Combined: ensand, herne, noon, ratoon, seeder, steen, torrid, toss

d. **For large data sets:**

allimages1	char-accuracy	word-accuracy	log-probability
OCR Model	58.393	11.197	-7.876
Transition Model	68.046	24.040	-7.175
Combined Model	70.839	31.489	-6.271

allimages2	char-accuracy	word-accuracy	log-probability
OCR Model	57.257	10.008	-7.874
Transition Model	67.689	24.177	-7.174
Combined Model	70.720	31.809	-6.271

allimages3	char-accuracy	word-accuracy	log-probability
OCR Model	57.258	9.917	-7.865
Transition Model	67.872	24.680	-7.167
Combined Model	70.629	31.946	-6.265

allimages4	char-accuracy	word-accuracy	log-probability
OCR Model	57.578	11.471	-7.869
Transition Model	68.248	24.680	-7.170
Combined Model	70.775	31.855	-6.267

allimages5	char-accuracy	word-accuracy	log-probability
OCR Model	58.531	11.563	-7.857
Transition Model	68.458	26.691	-7.158
Combined Model	71.068	33.318	-6.257

Extra Credit

1. When we scale the transition/ocr factors, it doesn't result in any improvement. Because we are modifying all the values and by formula, such factor cancels out in numerator and denominator.
2. When we try to scale the log probability (i.e raise the factors to some power), the character accuracy decreases

Model #0
Char Accuracy: 53.921569
Word Accuracy: 8.653846
Avg log prob : -7.808333

Model #1
Char Accuracy: 62.156863
Word Accuracy: 22.115385
Avg log prob : -6.669356

Model #2
Char Accuracy: 66.862745
Word Accuracy: 32.692308
Avg log prob : -5.860524

Above is the result of squaring the transition probabilities.

Model #0
Char Accuracy: 53.921569
Word Accuracy: 8.653846
Avg log prob : -7.808333

Model #1
Char Accuracy: 65.490196
Word Accuracy: 26.923077
Avg log prob : -7.408846

Model #2
Char Accuracy: 66.862745
Word Accuracy: 29.807692
Avg log prob : -6.588360

Above is result of taking square root of transition probability.

Moreover, change in skip factor doesn't change the overall accuracy much. But if we reduce the skip factor (to ~1), the accuracy decreases.