Learning Bayesian Network Structure from Heterogeneous Data

Shivkumar Shivaji
March 22, 2006

Structure Learning Techniques

- 2 General Approaches
  - Dependence Analysis posing learning as a constraint satisfaction problem. Proposed by Pearl in “A Theory of Inferred Causation” 91.
  - Search and Score Methods. A “score” describes each model's fitness. Bayesian and MDL are popular scores.

Search and Score

- NP-hard.
- Methods use the decomposability property.
- Basic idea:
  - Perform a series of arc changes one at a time.
  - Check whether resulting graph is a valid DAG.
  - Calculate scores before and after the change.
  - Acceptance depends on the difference between the scores

Subproject

- Classifying skill in intelligence games
- Both Go and Chess are applicable amongst many other games
- Don't you feel surprised when some people do not improve after years of playing a game?
- What does it take to be a successful go/chess player?
- Can Bayesian Networks help?

Chess

- I selected chess because I am not as familiar with go.
- I investigated the followed parameters with respect to success in training positions: age, math ability, mental speed, tournament experience, opening knowledge, middle game knowledge, endgame knowledge, time management, concentration ability, style, maturity, ability at speed chess.

Search Strategy

- Training sample size was 320 observations.
- Searchers – Simulated Annealing and Greedy Search
- Score Strategy – Examine all local moves, random local move, and Metropolis Hastings.

Performance of different Criteria

- Search about 50,000,000 node configurations.

Times in Minutes

	Simulated Annealing	Greedy
Random Local Move	10:00	9:32
All Local Moves	9:43	8:57
Metropolis	9:22	8:32

Performance Conclusions

- Not many local optima in solution – hence greedy was faster.
- Why Metropolis is the fastest searcher is less clear.

Result Accuracy

- Discretization of values lost accuracy.
- On testing with actual data – non training, the results seemed to show some correlations.
- Unfortunately, correlations could be accurately verified. Influence weights were not given by the program.
- About 80-90% of the correlations were accurate when I searched for counterexamples. For example, age and concentration were not directly correlated.

Other Ideas I tried

- Giving hints to the Bayesian structure to improve search speed. Search speed improved by about 4% for each connection hint. If we can partially locate the big correlations (i.e. 4 or more in bound nodes), then we save search time by 30% even if we make guesses on the hints!
- Parallel search – Independencies will greatly help here! Was successful when hints were given which aided full/partial partitioning.

Partitioning Algorithm

- Partition the graph into two relevant sets. Run structure learning on each set. Now apply structure learning on the aggregated graph with hints provided by the two graphs.
- Performance improvement - only 20-30%. My model was not sparse, and this clearly influenced the improvement potential. In some cases your results can be worse with parallelization!