Fast search for Dirichlet process mixture model

Hal Daume III

Presented by Yun Zhang

Oct 13 2011
Dirichlet Process Mixture Model

\[ G \mid \alpha, G_0 \sim \text{DP}(\alpha, G_0) \]
\[ \theta_n \mid G \sim G \]
\[ x_n \mid \theta_n \sim F(\theta_n) \]
Existing Algorithms

• Sampling methods – MCMC
  – Produce a true representation of the posterior
  – Convergence can be difficult to diagnose

• Variational techniques
  – Deterministic
  – Produce an approximation to the true posterior

  – Both of them run not very fast
Motivation

• Learning posteriors is expensive

• In some cases of DPMM, we only need to know about an approximate MAP cluster assignment for each observation, which lead to a limited search space

• We may need an efficient way of finding a high probability region as initial settings of MCMC
Problem setup

• Input: $x_{1:N}$

• Output: $c_{1:N}$, where:

$$c = \text{argmax } p(c, x)$$
A* Search Algorithm

• A greedy search algorithm which uses a distance-plus-cost heuristic function to determine the order of nodes to visit
  \[ g(x) = d(x) + h(x) \]

• An extension of Dijkstra’s algorithm

• One efficient implementation uses a heap (max-queue) to keep track of current searching progress
A* Search Algorithm

A max-heap

A* v.s. Dijkstra’s
Guarantees of global optima

- The heuristic function is “admissible”
  - The estimated cost must always be lower than or equal to the actual cost of reaching the goal state

- Heap size is unlimited
  - Search paths are never cut off
DP Search

- g: distance-plus-cost function
- b: heap size

```plaintext
function DPSearch
input: a scoring function g, beam size b, data x_{1:N}
output: a clustering c
1: initialize max-queue: Q ← [∅]
2: while Q is not empty do
3: remove state c_{1:N^0} from the front of Q
4: if N^0 = N then return c
5: for all clusters d in c and a new cluster do
6: let c^0 = c ⊕ ⟨d⟩
7: compute the score s = g(c^0, x)
8: update queue: Q ← Enqueue(Q, c^0, s)
9: end for
10: if b < ∞ and |Q| > b then
11: Shrink queue: Q ← Q_{1:b}
12: (drop lowest-scoring elements)
13: end if
14: end while
```
Scoring Function

\[ g(c^0, x) \geq \max p(c, x) \]
\[ g = \max p(c)p(x|c) \]
\[ = \max p(c)p(x|c) \]
\[ = \max p(c)g_{\text{Trivial}}(x|c^0)heu(x|c^0) \]

• Scoring function \( g() \) is an estimation of the posterior probability \( p(c, x) \)

• Our goal here is to find out a setting of cluster assignments \( c \) that can maximize the posterior probability
Maximize the prior $p(c)$

$$P(m \mid \alpha, N) = \frac{N!}{\alpha^N} \frac{\alpha \sum_{i=1}^{I} m_i}{\prod_{i=1}^{I} i^{m_i}(m_i!)}$$

- Coming from Chinese restaurant process
Trivial Scoring Function

$$g_{\text{Trivial}}(x \mid c^0) \triangleq \prod_{k \in c^0} H(x_{c^0=k})$$

- heuristic function is zero in log space
- A* becomes a Dijkstra’s algorithm, which lead to an inefficient search
Admissible Function

\[
g_{\text{Trivial}}(x \mid c^0) = \prod_{n=N^0+1}^{N} \max_{1 \leq k \leq K^0+1} \max_{\substack{c: \, c\mid N^0=c^0 \, \text{and} \, c_n=k}} H(x_n \mid x_{c_{1:n-1}=c_m})
\]  

(11)

- Treat unclustered data point independently

- For each unclustered data point \( x \), choose most like cluster label \( k \) for \( x \), then cluster remaining points as to only whether they fall into \( k \) or not

- Can be considered as admissible when a “replica” trick is used
Inadmissible Heuristic Function

\[ g_{\text{Inad}}(x \mid c^0) \triangleq g_{\text{Trivial}}(x \mid c^0) \prod_{n= N^0 + 1}^{N} H(x_n) \]

• Use the marginal likelihood as heuristic, which means for each unclustered data point, assign them a new cluster number

• No longer overestimate the posterior probability, therefore not admissible
Experiment 1: Artificial data (DPGMM)
Experiment 2: Handwritten data

• Dirichlet/Multinomial setting

• Conclusion is our search algorithm runs much faster than Gibbs sampler, and gives better MAP estimation of cluster assignments.
Experiment 3: NIPS documents

- Still Dirichlet/Multinomial setting

- Conclusion is our search algorithm runs much much faster than Gibbs sampler
Advantages & Limitations

- Fast to find a MAP cluster assignment for each data point

- Cannot represent the true posterior

- Applies only to exponential families with conjugate prior (or at least speed will be slowed down when apply to non-conjugate distributions)