Fall 2014

University of Southern California

December 22, 2014

Summary

- TA for Machine Learning (50%)
- Two classes
- Meeting...meetings...
- Submitted one paper with Max
- Research project with Fei (collaboration with Yuan)
- Research project with Yan

Scalable inference for overlapping communities

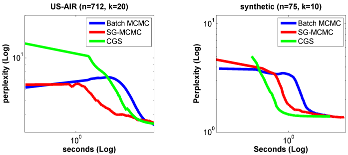

Figure: Batch MCMC vs SG-MCMC vs CGS

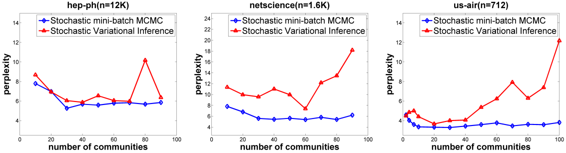

Figure: SG-MCMC vs SVI

- Submitted to AISTATS 2015
- Extended version(with 1.8B edges) may be submitted to KDD 2015

Memory Efficient Bayesian Deep Factored Mixed Membership Models

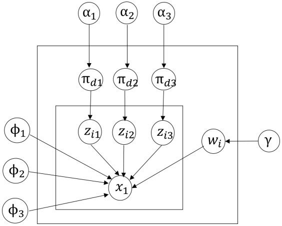

Figure: 3-layer factored LDA

For each document d,

for each layer l

draw (sub)topic distribution π_{dl}

for each word i

draw (sub)topic z_{il} for $l \in \{1, .., L\}$

draw word x_i from convex combination of $\{\phi_l\}$

Memory Efficient Bayesian Deep Factored Mixed Membership Models(cont.)

Some benefits:

- Suppose 3-layer model, each layer has L_1, L_2, L_3 components, this can represent $\prod_{i=1}^{3} L_i$ mixtures (exponential).
- Less parameters comparing to shallow models.
- Less memory. (becomes very obvious once we have k equals to hundreds of thousand, uses logarithm amount of space comparing to original model)
- Some hierarchical structures

Learning:

- (Stochastic) variational inference, fixed point iteration, L-BFGS
- Or propose better inference

Progress:

Coding, Debugging.... (also need to include deep factored MMSB)

Parallel Stochastic Variational Inference for Bayesian Tensor Factorization

Algorithm 1 Stochastic variational inference

Initialize $U, V, W, \alpha, \beta, \gamma$
while not converge **do**
  Sample a mini-batch of entries, Ω_t, from X
  for $i \in \Omega_t^i$ **do**
    for k=1,...K **do**
      update s_{ik}^u, u_{ik} using (8),(9)
    end for
  end for
  for $j \in \Omega_t^j$ **do**
    for k=1,...K **do**
      update s_{jk}^v, v_{jk} using (10),(11)
    end for
  end for
  for $l \in \Omega_t^l$ **do**
    for k=1,...K **do**
      update s_{lk}^w, w_{lk} using (12),(13)
    end for
  end for
  update α, β, γ using (14),(15),(16)
  update τ
end while

Algorithm 2 Parallel Inference

Initialize sets $S^u = [1, ..., I], S^v = [1, ..., J], S^w = [1, ..., L]$
Initialize mini-batch size for each dimension $N^u = |S^u|/K, N^v = |S^v|/K, N^l =$
  where K is a constant that large than the number of threads T
while not converge AND not reach max iteration **do**
  for $t = 1, ..., T$ parallel **do**
    sample mini-batch of indexes I_u^t, I_v^t, I_w^t from S^u, S^v, S^w, the size of mini-b
    erned by N^u, N^v, N^l. Then remove I_u^t, I_v^t, I_w^t from sets S^u, S^v, S^w
    Form subset of tensor Ω_t by I_u^t, I_v^t, I_w^t
    Update parameters as in Algorithm 1.
    Add I_u^t, I_v^t, I_w^t back into the index sets.
  end for
end while

Extension: distributed settings

Learning from Ordinal Data

- hmm.. this is a hard problem...
- need a simple, elegant solution....
- Working hard on experiments, will tell you more about this.

Plan for winter

- Coding :)
- Debugging :(
- Getting results :)