Scalable Machine Learning CS281B, STAT241B — May 1, 2012
http://alex.smola.org/teaching/berkeley2012 — due on May 11, 2012
UC Berkeley — Alexander J. Smola

Problem Set 5 (backup set) — Distributed Inference

---

1 Distributed Synchronization

When synchronizing updates between computers one may use distributed caching. Your task is to prove that the updates are consistent. That is, you need to show that local changes are accurately distributed and aggregated globally. For simplicity we consider a single variable x which is being modified by n machines.

Denote by x the global variable. Moreover, let x_i, x_i' for $i \in \{1, \dots n\}$ be the machine local copies. The system starts off by setting $x = x_i = x_i'$ for all i. Moreover, we set $s_i = \text{FALSE}$ (message pending). For some time the variables x_i are being updated by δ_{ij}. That is, we update

$$x_i \leftarrow x_i + \delta_{ij} \tag{1}$$

for all j sequentially (note that there is no need that the same number of updates occurs at different machines). To keep matters synchronized we execute the following protocol:

LocalSend(i)
if $s_i = \text{FALSE}$ **then**
  Lock x_i and x_i' (prevent updates during message generation)
  $\eta \leftarrow x_i - x_i'$
  $x_i' \leftarrow x_i$
  Unlock x_i and x_i'
  GlobalReceive(η)
  $s_i \leftarrow \text{TRUE}$
end if

LocalReceive(x)
Lock x_i and x_i' (prevent updates during message generation)
$\eta \leftarrow x - x_i'$
$x_i \leftarrow x_i + \eta$
$x_i' \leftarrow x$
$s_i \leftarrow \text{FALSE}$
Unlock x_i and x_i'

GlobalSend(i)
LocalReceive(x) to machine i

GlobalReceive(η)
$x \leftarrow x + \eta$

1.1 Prove that after all updates are completed the algorithm will converge to

$$x = x_{\text{original}} + \sum_{i,j} \delta_{ij} \tag{2}$$

and moreover $x_i = x_i' = x$ for all i. That is, the algorithm is correct and convergent.

1.2 What could happen if we did not use s_i. Hint — consider the case where $s_i = \text{TRUE}$. What does this mean in terms of messages queued up in GlobalReceive?

1.3 Why do we need to lock x_i?

Problem Set 5 (backup set) — Distributed Inference

2 Bayesian Nonparametric Clustering

The Dirichlet Process, and its generalization, the Pitman-Yor process define distributions over distributions containing a countable number of atoms. These tools are useful, e.g. whenever we want to partition objects into a countable number of groups but do not necessarily want to specify how many we have, yet at the same time, we would like to specify the distribution from which these .

Denote by H some probability measure on a domain $\mathcal{X}$ and let $\alpha > 0$. Then the Dirichlet Process $DP(H, \alpha)$ is defined as follows: for any partitioning of $\mathcal{X}$ into k sets $\mathcal{X}_i$ with $\cup_{i=1}^k \mathcal{X}_i = \mathcal{X}$ and $\mathcal{X}_i \cap \mathcal{X}_j = \emptyset$ for $i \neq j$ we have that draws from $DP(H, \alpha)$ satisfy

$$(X(\mathcal{X}_1), \ldots, X(\mathcal{X}_k)) \sim \text{Dirichlet}(\alpha H(\mathcal{X}_1), \ldots, H(\mathcal{X}_k)). \tag{3}$$

In other words, at a resolution given by the sets $\mathcal{X}_i$ the measure behaves like a Dirichlet distribution with associated weights $\alpha H(\mathcal{X}_i)$.

2.1 Expected Number of Items for a Dirichlet Process

There exists an equivalent representation in terms of the Chinese Restaurant Process which works as follows: for each $x_n \sim DP(H, \alpha)$ we draw $x_{n+1} \sim \text{Unif}(x_1, \ldots, x_n)$ with probability $\frac{n}{n+\alpha}$ and with probability $\frac{\alpha}{n+\alpha}$ we draw it from H.

Compute a tight upper and lower bound of the number of distinct elements after n draws from $DP(H, \alpha)$. Hint — it is OK to approximate sums by integrals. Hint — the expected number is logarithmic in n.

2.2 Expected Number of Items for a Pitman Yor Process

The Pitman-Yor process allows for larger numbers of distinct items by boosting the probability of new objects beyond $\frac{\alpha}{n+\alpha}$. The key difference to the Dirichlet process is that now, instead of assigning previously drawn items the probability $\frac{n_i}{n+\alpha}$ we assign them the probability $\frac{n_i - d}{n+\alpha}$ and correspondingly the probability of a draw from H has probability $\frac{\alpha + md}{n+\alpha}$. Here m is the number of items drawn from H so far and $d \in (0, 1)$ is the discount factor.

Argue that $m = O(n^d)$. Hint — it is sufficient if you can show that the rate cannot be lower or higher than $O(n^d)$ by considering the expected case and by approximating m, n to be continuous variables. Note that this procedure provides a power law behavior.

For a nice application to language modeling see http://jmlr.csail.mit.edu/papers/volume12/goldwater11a/goldwater11a.pdf.

Problem Set 5 (backup set) — Distributed Inference

3 Semi-Markov Model

Markov models are good when it comes to modeling processes where there is a fair amount of switching between states or alternatively where the length of a segment is exponentially distributed. Different transition behavior can, in principle, be encoded by much longer dependencies, but the computational cost for performing inference in this model is exponential in the length of the context.

Semi-Markov Models address this issue by modifying the transition probability between states. Instead of having a transition probability $p(i|j)$ for transitioning from state j at time t to state i at time $t+1$ we use a model of distributions over states and segments. That is, assume that for a time interval $[0, T]$ we have pairs (t_i, x_i) to denote the times a particular state is reached. Moreover assume that $t_0 = 0$ and $t_n = T$ (for some unknown n) and that $t_i < t_{i+1}$. Finally assume that $x_i \neq x_{i+1}$ for all i. In this case we may express the sequence likelihood by

$$p((t_0, x_0), \dots (T, x_n)) \propto \psi(x_0) \cdot \prod_{i=1}^{n} \psi(x_i, (t_i - t_{i-1})|x_{i-1}) \tag{4}$$

for some function ψ.

3.1 Under the assumption that each segment cannot be too long, i.e. that $t_i - t_{i-1} \leq c$ for some $c \in \mathbb{N}$ design an algorithm to normalize (4). What is the computational complexity.

Hint — note that each segment only depends on the position and state of the previous segment. Use this to generate a recursion. Hint — to bound computational complexity you may assume that each segment has a maximum length τ.

3.2 Derive an algorithm for finding the most likely sequence of annotations. Hint — you can replace a semiring.

Problem Set 5 (backup set) — Distributed Inference

4 Document Clustering

Assume that we are given a collection of documents, drawn from a mixture of multinomial distributions. That is, we assume the following model:

- The distribution over clusters is multinomial and the latter is smoothed by a Dirichlet, that is

$$y_i \sim \text{Multinomial}(\pi) \tag{5}$$
$$\pi \sim \text{Dirichlet}(\alpha) \tag{6}$$

- For each cluster $i \in \{1..k\}$ the word distribution is drawn from

$$\psi \sim \text{Dirichlet}(\beta) \tag{7}$$

- For each document x_i we draw all associated words from a multinomial distribution, that is

$$x_{ij} \sim \text{Multinomial}(\psi_{y_i}) \tag{8}$$

  In other words, y_i selects the multinomial distribution.

For computational convenience we do not attempt to model the document length explicitly.

4.1 Sampling algorithm

Derive a Gibbs sampling algorithm to draw from the model. For this purpose you need to derive the following.

1. Integrate out π and ψ in the same fashion as the collapsed Gibbs sampler in LDA.
2. Document likelihood $p(x_i)$
3. Conditional cluster likelihood $p(y_i|x_i)$

4.2 Implementation

Use the Reuters newswire dataset to group documents into $k = 100$ clusters.

- Documents are at http://kdd.ics.uci.edu/databases/reuters21578/reuters21578.html.
- Perform 1000 Gibbs sampling iterations for convergence.
- Compute the document average log-likelihood and plot it as a function of the number of iterations.