Statistical Learning Models for Text and Graph Data
Topic Models

Yangqiu Song

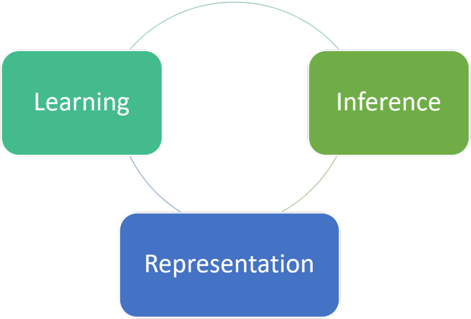
Hong Kong University of Science and Technology

yqsong@cse.ust.hk

October 25, 2019

∗Contents are based on materials created by David Mackay (Introduction to Monte Carlo methods, 1998)

Course Organization



- Representation: language models, word embeddings, topic models, knowledge graphs
- Learning: supervised learning,unsupervised learning, semi-supervised learning, distant supervision, indirect supervision, sequence models, deep learning, optimization techniques
- Inference: constraint modeling, joint inference, search algorithms

Overview

1. Language Models: Recap
2. Topic Models
3. Probabilistic Latent Semantic Analysis (PLSA)
4. Latent Dirichlet Allocation (LDA)
    - Motivation: Bayesian Modeling
    - Background of Monte Carlo Methods
        - Importance Sampling
        - Rejection Sampling
        - Metropolis Methods
        - Gibbs Sampling
        - Sampling for EM Algorithm
    - Collapsed Gibbs Sampling for LDA

Overview

1. Language Models: Recap
2. Topic Models
3. Probabilistic Latent Semantic Analysis (PLSA)
4. Latent Dirichlet Allocation (LDA)
    - Motivation: Bayesian Modeling
    - Background of Monte Carlo Methods
        - Importance Sampling
        - Rejection Sampling
        - Metropolis Methods
        - Gibbs Sampling
        - Sampling for EM Algorithm
    - Collapsed Gibbs Sampling for LDA

Bayesian Inference

- Suppose we have a Basysian learning problem $P(\Theta|X) \propto P(X|\Theta)P(\Theta)$ $(X = \{x_1, \ldots, N\})$
- If we want to predict for a new coming data x
- Maximum a posterior (MAP) makes a point estimation $\Theta^* = \max_\Theta P(\Theta|X)$, and makes a prediction as $P(x|\Theta^*)$
- Full Bayesian uses $P(x|X) = \int_\Theta P(x|\Theta)P(\Theta|X)d\Theta$
- In general, we have a lot of following cases need to be estimated:

$$\mathbb{E}_{P(\mathbf{x})}[\phi(\mathbf{x})] = \int_{\mathbf{x}} \phi(\mathbf{x})P(\mathbf{x})d\mathbf{x}$$

- One way to solve this (especially when $P(\mathbf{x})$ is difficult to compute) is using sampling:

$$\hat{\mathbb{E}}_{P(\mathbf{x})}[\phi(\mathbf{x})] = \frac{1}{R}\sum_{\mathbf{x}^{(r)} \sim P(\mathbf{x})}^{R} \phi(\mathbf{x}^{(r)})$$

Sampling

$$\Phi = \mathbb{E}_{P(\mathbf{x})}[\phi(\mathbf{x})] = \int_{\mathbf{x}} \phi(\mathbf{x}) P(\mathbf{x}) d\mathbf{x}$$

- We call $P(\mathbf{x})$ the target density
- We assume $\mathbf{x}$ is a $\mathbb{R}^d$ vector with real/discrete components $\mathbf{x}^i$
- We concentrate on the sampling problem, because if we have solved it, then we can solve the expectation problem by

$$\hat{\Phi} = \hat{\mathbb{E}}_{P(\mathbf{x})}[\phi(\mathbf{x})] = \frac{1}{R} \sum_{\mathbf{x}^{(r)} \sim P(\mathbf{x})}^{R} \phi(\mathbf{x}^{(r)})$$

- The expectation of $\hat{\Phi}$ is Φ
- The variance of $\hat{\Phi}$ will decrease as $\frac{\sigma^2}{R}$ where σ^2 is the variance of Φ:

$$\sigma^2 = \int_{\mathbf{x}} [\phi(\mathbf{x}) - \Phi]^2 P(\mathbf{x}) d\mathbf{x}$$

  which means the accuracy of the sampling is independent of the dimensionality of the space sampled

  - A few as a dozen independent samples $\mathbf{x}^{(r)}$ suffice of estimate Φ satisfactorily

However, why is sampling from $P(\mathbf{x})$ hard?

- We assume that the density from which we wish to draw samples $P(\mathbf{x})$ can be evaluated, at least to with a multiplicative constant:

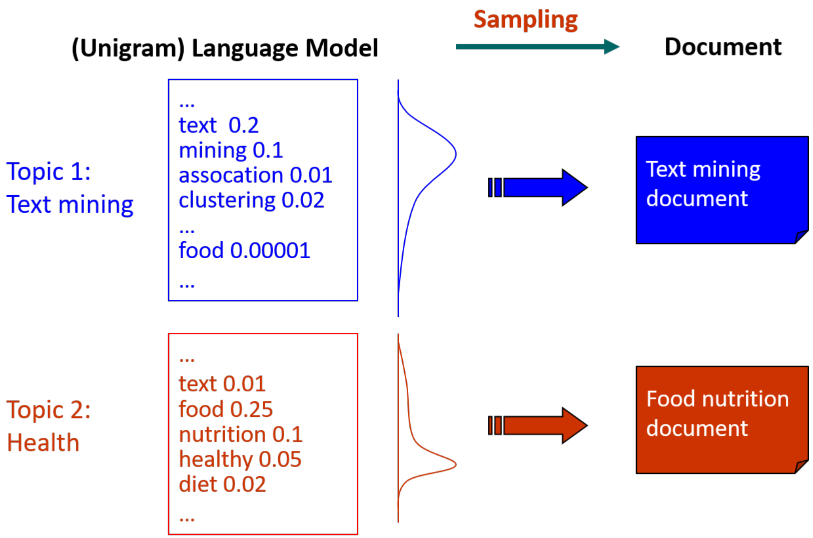
$$P(\mathbf{x}) = P^*(\mathbf{x})/Z$$

- If we can evaluate $P^*(\mathbf{x})$, why can we not easily obtain Φ?
  - We do not know the normalizing constant

$$Z = \int_{\mathbf{x}} d\mathbf{x} P^*(\mathbf{x})$$

  - Even if we know Z, drawing samples from $P(\mathbf{x})$ is still challenging, especially in high-dimensional spaces

Recall: Generative View of Text Documents



(Unigram) Language Model

Sampling

Document

Topic 1:
Text mining

…
text 0.2
mining 0.1
assocation 0.01
clustering 0.02
…
food 0.00001
…

Text mining document

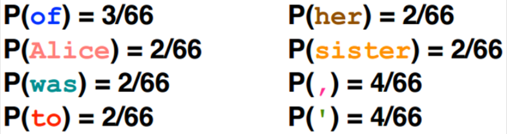
Topic 2:
Health

…
text 0.01
food 0.25
nutrition 0.1
healthy 0.05
diet 0.02
…

Food nutrition document

Generating Text from Language Models

Example



P(of) = 3/66

P(Alice) = 2/66

P(was) = 2/66

P(to) = 2/66

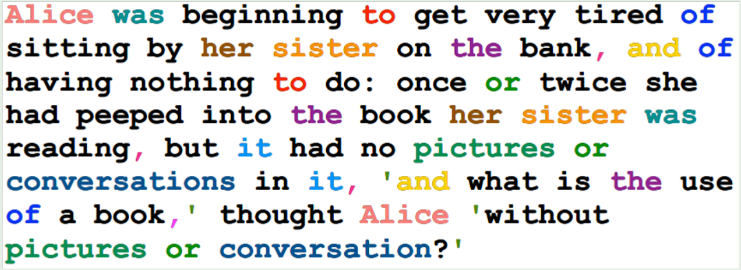
P(her) = 2/66

P(sister) = 2/66

P(,) = 4/66

P(') = 4/66

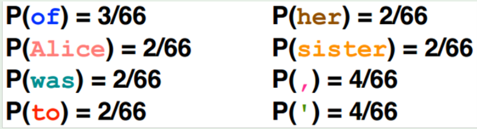
Under a unigram language model:



Alice was beginning to get very tired of
sitting by her sister on the bank, and of
having nothing to do: once or twice she
had peeped into the book her sister was
reading, but it had no pictures or
conversations in it, 'and what is the use
of a book,' thought Alice 'without
pictures or conversation?'

Generating Text from Language Models

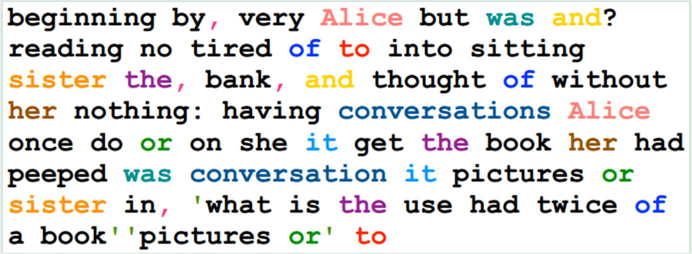
Example



P(of) = 3/66	P(her) = 2/66
P(Alice) = 2/66	P(sister) = 2/66
P(was) = 2/66	P(,) = 4/66
P(to) = 2/66	P(') = 4/66

Under a unigram language model:

The same likelihood!



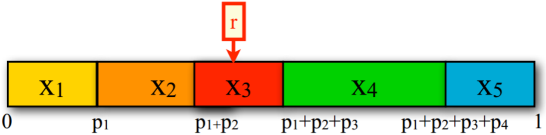
beginning by, very Alice but was and?
reading no tired of to into sitting
sister the, bank, and thought of without
her nothing: having conversations Alice
once do or on she it get the book her had
peeped was conversation it pictures or
sister in, 'what is the use had twice of
a book''pictures or' to

Recall: Computer Simulation

Sample from a discrete distribution $P(X)$, assuming n outcomes in the event space X

Algorithm 1 Sample from a distribution $P(X)$

```
for $t = 1$ to $T$ do
    Divide the interval $[0, 1]$ into $n$ intervals according to the probabilities of the outcomes
    Generate a random number $r$ between 0 and 1
    Return $x_i$ where $r$ falls into $[\sum_0^{i-1} p_i, \sum_0^{i} p_i]$
end for
```



Uniform Sampling

- Having agreed that we cannot visit every location in the space, we might consider trying to solve sampling by uniform sampling:
  - Sample $\mathbf{x}^{(r)}$ uniformly and evaluate $P^*(\mathbf{x}^{(r)})$ to give

$$Z_R = \sum_{r=1}^{R} P^*(\mathbf{x}^{(r)})$$

  - and estimate $\Phi = \mathbb{E}_{P(\mathbf{x})}[\phi(\mathbf{x})] = \int_{\mathbf{x}} \phi(\mathbf{x}) P(\mathbf{x}) d\mathbf{x}$ by

$$\hat{\Phi} = \sum_{r=1}^{R} \phi(\mathbf{x}^{(r)}) \frac{P^*(\mathbf{x}^{(r)})}{Z_R}$$

- Is there anything wrong with this strategy?

Is there anything wrong with this strategy?

- Let's assume $\phi(\mathbf{x})$ is a benign, smoothly varying function, and concentrated on the nature of $P^*(\mathbf{x})$
- A high dimensional distribution is often concentrated in a small region of the state space known as its typical set T
  - whose volume is given by $|T| \simeq 2^{H(\mathbf{X})}$
  - $H(\mathbf{X})$ is the entropy of the probability distribution $H(\mathbf{X}) = \sum_{\mathbf{x}} P(\mathbf{x}) \log_2 \frac{1}{P(\mathbf{x})}$
- $\Phi = \int_{\mathbf{x}} \phi(\mathbf{x}) P(\mathbf{x}) d\mathbf{x}$ will be principally determined by values in typical set
- If we have d random variables with binary values, the total size of state space is 2^d and the typical set size is 2^H
  - Each sample has a chance $2^H/2^d$ of falling into typical set
  - We need $R_{\min} \simeq O(2^{d-H})$ samples

One Dimensional Sampling Example

- Consider $P^* = \exp\{0.4(x-0.4)^2 - 0.08x^4\}$, $x \in (-\infty, \infty)$
- To give a simpler problem, we can discretize the variable x and ask for samples from the discrete prob.

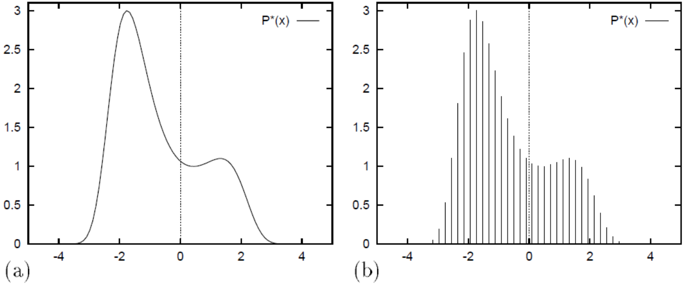


Figure 1. (a) The function $P^*(x) = \exp\left[0.4(x-0.4)^2 - 0.08x^4\right]$. How to draw samples from this density? (b) The function $P^*(x)$ evaluated at a discrete set of uniformly spaced points $\{x_i\}$. How to draw samples from this discrete distribution?

- There are 50 uniformly spaced points in one dimension

The Cost of Computing Z

- To compute Z, we have to visit every point in the space

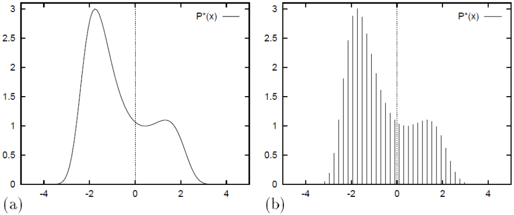


Figure 1. (a) The function $P^*(x) = \exp\left[0.4(x-0.4)^2 - 0.08x^4\right]$. How to draw samples from this density? (b) The function $P^*(x)$ evaluated at a discrete set of uniformly spaced points $\{x_i\}$. How to draw samples from this discrete distribution?

- If we evaluate $p_i^* = P^*(x_i)$ at each point x_i, we can compute $Z = \sum_i p_i^*$ and $p_i = p_i^*/Z$
- If our system had $d = 1000$ dimensions of binary variables
  - Then the corresponding number of points would be 2^{1000}

Overview

1. Language Models: Recap
2. Topic Models
3. Probabilistic Latent Semantic Analysis (PLSA)
4. Latent Dirichlet Allocation (LDA)
   - Motivation: Bayesian Modeling
   - Background of Monte Carlo Methods
     - Importance Sampling
     - Rejection Sampling
     - Metropolis Methods
     - Gibbs Sampling
     - Sampling for EM Algorithm
   - Collapsed Gibbs Sampling for LDA

Importance Sampling

- Importance sampling is not a method for generating samples from $P(\mathbf{x})$
- It is just a method for estimating the expectations of a function $\phi(\mathbf{x})$
- Let's imagine the target distribution is a one-dimensional density $P(x)$

$$P(x) = \frac{P^*(x)}{Z}$$

  but $P(x)$ is too complicated to sample from directly
- We assume $Q(x) = \frac{Q^*(x)}{Z_Q}$ is a simpler density from which we can generate samples
- In importance sampling, we generate R samples $\{x^{(r)}\}_{r=1}^R$ from $Q(x)$
- Then Φ can be estimated by

$$\hat{\Phi} = \frac{\sum_r w_r \phi(x^{(r)})}{\sum_r w_r}$$

where $w_r = \frac{P^*(x^{(r)})}{Q^*(x^{(r)})}$

Importance Sampling: A Toy Example

- Φ can be estimated by

$$\hat{\Phi} = \frac{\sum_r w_r \phi(x^{(r)})}{\sum_r w_r}$$

where $w_r = \frac{P^*(x^{(r)})}{Q^*(x^{(r)})}$

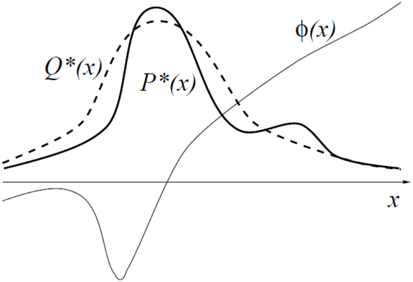


Figure 2. Functions involved in importance sampling. We wish to estimate the expectation of $\phi(x)$ under $P(x) \propto P^*(x)$. We can generate samples from the simpler distribution $Q(x) \propto Q^*(x)$. We can evaluate Q^* and P^* at any point.

Importance Sampling: A Toy Example

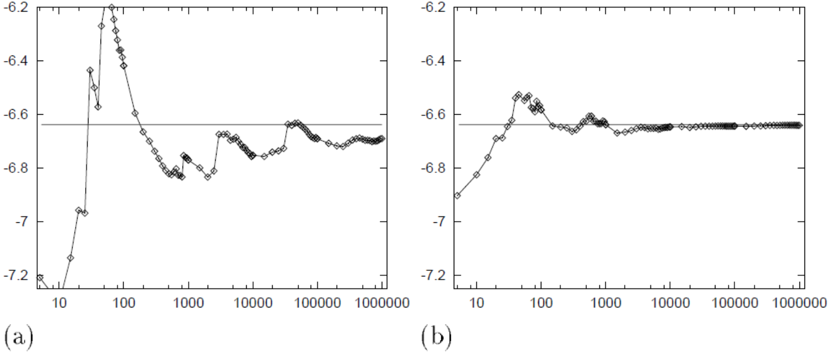


Figure 3. Importance sampling in action: a) using a Gaussian sampler density; b) using a Cauchy sampler density. Horizontal axis shows number of samples on a log scale. Vertical axis shows the estimate $\hat{\Phi}$. The horizontal line indicates the true value of Φ.

Importance Sampling in Many Dimensions

- Importance sampling suffers from two difficulties
  - We clearly need to obtain samples that lie in the typical set
  - Even if we obtain samples in the typical set, the weights associated with theose samples are likely to vary by large factors

Overview

1. Language Models: Recap
2. Topic Models
3. Probabilistic Latent Semantic Analysis (PLSA)
4. Latent Dirichlet Allocation (LDA)
    - Motivation: Bayesian Modeling
    - Background of Monte Carlo Methods
        - Importance Sampling
        - Rejection Sampling
        - Metropolis Methods
        - Gibbs Sampling
        - Sampling for EM Algorithm
    - Collapsed Gibbs Sampling for LDA

Rejection Sampling

- We again assume one dimensional complicated density $P(x) = \frac{P^*(x)}{Z}$
- We assume a simpler proposal density $Q(x)$ which we can evaluate and can generate samples from
- We further assume for all x, $cQ^*(x) > P^*(x)$

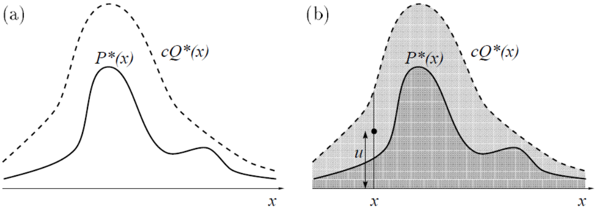


Figure 4. Rejection sampling. a) The functions involved in rejection sampling. We desire samples from $P(x) \propto P^*(x)$. We are able to draw samples from $Q(x) \propto Q^*(x)$, and we know a value c such that $cQ^*(x) > P^*(x)$ for all x. b) A point (x, u) is generated at random in the lightly shaded area under the curve $cQ^*(x)$. If this point also lies below $P^*(x)$ then it is accepted.

Rejection Sampling

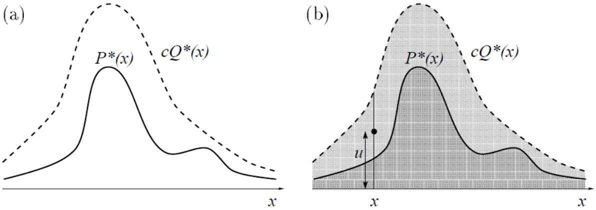


Figure 4. Rejection sampling. a) The functions involved in rejection sampling. We desire samples from $P(x) \propto P^*(x)$. We are able to draw samples from $Q(x) \propto Q^*(x)$, and we know a value c such that $cQ^*(x) > P^*(x)$ for all x. b) A point (x, u) is generated at random in the lightly shaded area under the curve $cQ^*(x)$. If this point also lies below $P^*(x)$ then it is accepted.

- We first generate x from $Q(x)$
- We evaluate $cQ^*(x)$ and generate a uniformly distributed variable from the interval $[0, cQ^*(x)]$
- We then evaluate $P^*(x)$ and accept or reject the sample x by comparing u with $P^*(x)$
    - If $u > P^*(x)$ then x is rejected

Rejection Sampling

- Reject sampling will work best when Q is a good proximation of P
- c grows exponentially with the dimensionality N (MacKay (1998))
- While it is a useful method for one-dimensional problems, it is not a practical technique for high-dimensional distributions $P(\mathbf{x})$

Overview

1. Language Models: Recap
2. Topic Models
3. Probabilistic Latent Semantic Analysis (PLSA)
4. Latent Dirichlet Allocation (LDA)
   - Motivation: Bayesian Modeling
   - Background of Monte Carlo Methods
     - Importance Sampling
     - Rejection Sampling
     - Metropolis Methods
     - Gibbs Sampling
     - Sampling for EM Algorithm
   - Collapsed Gibbs Sampling for LDA

Motivation

- Importance sampling and rejection sampling only work well if the proposal density $Q(x)$ is similar to $P(x)$
- In large and complex problems, it is difficult to create a single density $Q(x)$ that has this property
- The Metropolis algorithm makes use of a proposal density $Q(x', x^{(t)})$ which depends on the current state $x^{(t)}$
  - It is not necessarily for $Q(x', x^{(t)})$ to look at all similar to $P(x)$

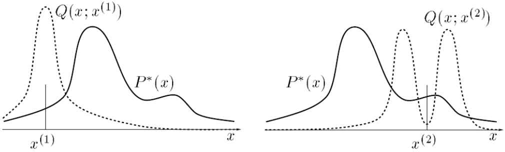


Figure 6. Metropolis method in one dimension. The proposal distribution $Q(x'; x)$ is here shown as having a shape that changes as x changes, though this is not typical of the proposal densities used in practice.

Metropolis Method

- We again assume that we can evaluate $P^*(x)$ for any x
- A tentative new x' is generated from the proposal density $Q(x'|x^{(t)})$
- To decide whether to accept the new state, we compute the quantity:

$$a = \frac{P^*(x')Q(x^{(t)}|x')}{P^*(x^{(t)})Q(x'|x^{(t)})}$$

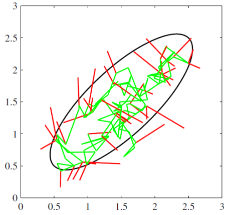
  - If $a \geq 1$, then accept the new x'
  - Otherwise, the new state is accepted with probability a

  - If the state is accepted, we set $x^{(t+1)} = x'$
  - If the state is rejected, we set $x^{(t+1)} = x^{(t)}$

- This is different from rejection sampling: in Metropolis method, a rejection causes the current state sent to the generated list another time
- The samples in a Metropolis simulation of T iterations are correlated

Variants

$$a = \frac{P^*(x')Q(x^{(t)}|x')}{P^*(x^{(t)})Q(x'|x^{(t)})}$$



Steps that are accepted are shown as green lines, and rejected steps are shown in red.

- When $Q(x^{(t)}, x') = Q(x', x^{(t)})$, it is called Metropolis algorithm (Metropolis et al., 1953)
- When $Q(x^{(t)}, x') \neq Q(x', x^{(t)})$, it is know as Metropolis-Hastings algorithm (Hastings, 1970)
- Metropolis methods are know as Markov chain Monte Carlo methods

Markov Chains

- A first order Markov chain is defined to be a series of random variables $x^{(1)}, \ldots, x^{(M)}$ such that the following conditional independence property holds

$$P(x^{(t+1)}|x^{(1)}, \ldots, x^{(t)}) = P(x^{(t+1)}|x^{(t)})$$

  where we define the transition probability $T(x^{(t)}, x^{(t+1)}) = P(x^{(t+1)}|x^{(t)})$

  - A Markov chain is called homogeneous if the transition probabilities are the same for all t

- The marginal probability for a particular variable can be expressed in terms of the marginal probability for the previous variable in the chain in the form

$$P(x^{(t+1)}) = \sum_{x^{(t)}} P(x^{(t+1)}|x^{(t)})P(x^{(t)})$$

Markov Chains (Cont'd)

- A distribution is said to be invariant, or stationary, with respect to a Markov chain if each step in the chain leaves that distribution invariant
- For a homogeneous Markov chain $P(z)$ is invariant if

$$P(x) = \sum_{x'} T(x', x) P(x')$$

- A sufficient (but not necessary) condition for ensuring that the required distribution $P(x)$ is invariant is to choose the transition probabilities to satisfy the property of detailed balance, defined by

$$P(x) T(x, x') = P(x') T(x', x)$$

- It is easy to verify that

$$\sum_{x'} T(x', x) P(x') = \sum_{x'} P(x) T(x, x') = P(x) \sum_{x'} T(x, x') = P(x)$$

Metropolis methods are know as Markov chain Monte Carlo methods

- From the probability to accept a new state:

$$a(x, x^{(t)}) = \min\left(1, \frac{P^*(x')Q(x^{(t)}|x')}{P^*(x^{(t)})Q(x'|x^{(t)})}\right)$$

- We have the joint probability of two consecutive states as

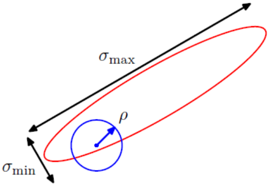
$$\begin{aligned} & P^*(x^{(t)}) \cdot Q(x'|x^{(t)})a(x, x^{(t)}) \\ = \; & \min(P^*(x^{(t)})Q(x'|x^{(t)}), P^*(x')Q(x^{(t)}|x')) \\ = \; & \min(P^*(x')Q(x^{(t)}|x'), P^*(x^{(t)})Q(x'|x^{(t)})) \\ = \; & P^*(x') \cdot Q(x^{(t)}|x')a(x^{(t)}, x') \end{aligned}$$

  as required

- Metropolis method actually samples from the required distribution $P(x)$

Sampling Effects

- The specific choice of proposal distribution can have a marked effect on the performance of the algorithm



- The scale ρ of the proposal distribution should be on the order of the smallest standard deviation σ_{min}
- The iteration T should be at least $(\sigma_{\text{max}}/\sigma_{\text{min}})^2$ to obtain approximately independent samples

Simulation of Sampling

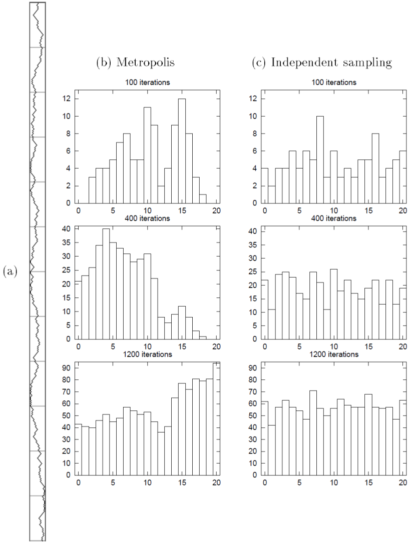


Figure 8. Metropolis method for a toy problem. (a) The state sequence for $t = 1 \ldots 600$. Horizontal direction = states from 0 to 20; vertical direction = time from 1 to 600; the cross bars mark time intervals of duration 50. (b) Histogram of occupancy of the states after 100, 400 and 1200 iterations. (c) For comparison, histograms resulting when successive points are drawn *independently* from the target distribution.

$$P(x) = \begin{cases} \frac{1}{21} & x \in \{0, 1, \ldots, 20\} \\ 0 & \textit{otherwise} \end{cases}$$

$$Q(x'|x) = \begin{cases} \frac{1}{2} & x' = x \pm 1 \\ 0 & \textit{otherwise} \end{cases}$$

- Rejection will occur only when the proposal takes the state $x' = -1$ or $x' = 21$
- It takes $\approx T^2 = 100$ (178) iterations to reach 0 or 20
- It takes ≈ 400 (540) iterations to reach both 0 and 20

Overview

1. Language Models: Recap

2. Topic Models

3. Probabilistic Latent Semantic Analysis (PLSA)

4. Latent Dirichlet Allocation (LDA)
   - Motivation: Bayesian Modeling
   - Background of Monte Carlo Methods
     - Importance Sampling
     - Rejection Sampling
     - Metropolis Methods
     - Gibbs Sampling
     - Sampling for EM Algorithm
   - Collapsed Gibbs Sampling for LDA

Gibbs Sampling

- In the general case of a system with K variables, a single iteration involves sampling one parameter at a time:
  - $x_1^{(t+1)} \sim P(x_1 | x_2^{(t)}, x_3^{(t)}, \ldots, x_K^{(t)})$
  - $x_2^{(t+1)} \sim P(x_2 | x_1^{(t+1)}, x_3^{(t)}, \ldots, x_K^{(t)})$
  - $x_3^{(t+1)} \sim P(x_3 | x_1^{(t+1)}, x_2^{(t+1)}, \ldots, x_K^{(t)})$
  - ...
  - $x_K^{(t+1)} \sim P(x_K | x_1^{(t+1)}, x_2^{(t+1)}, \ldots, x_{K-1}^{(t+1)})$
- Denote $\mathbf{x}_{\setminus k}^{(t)} = \{x_1^{(t+1)}, x_2^{(t+1)}, \ldots, x_{k-1}^{(t+1)}, x_{k+1}^{(t)}, \ldots, x_K^{(t)}\}$
- Gibbs sampling can be viewed as a Metropolis method

$$\begin{aligned} a_G &= \frac{P^*(\mathbf{x}')Q(\mathbf{x}^{(t)}|x')}{P^*(\mathbf{x}^{(t)})Q(\mathbf{x}'|\mathbf{x}^{(t)})} = \frac{P(\mathbf{x}')P(x_k^{(t)}|\mathbf{x}'_{\setminus k})}{P(\mathbf{x}^{(t)})P(x'_k|\mathbf{x}^{(t)}_{\setminus k})} \\ &= \frac{P(x'_k|\mathbf{x}'_{\setminus k})P(\mathbf{x}'_{\setminus k})P(x_k^{(t)}|\mathbf{x}'_{\setminus k})}{P(x_k^{(t)}|\mathbf{x}^{(t)}_{\setminus k})P(\mathbf{x}^{(t)}_{\setminus k})P(x'_k|\mathbf{x}^{(t)}_{\setminus k})} \overset{\mathbf{x}'_{\setminus k}=\mathbf{x}^{(t)}_{\setminus k}}{=} \frac{P(x'_k|\mathbf{x}'_{\setminus k})P(\mathbf{x}'_{\setminus k})P(x_k^{(t)}|\mathbf{x}'_{\setminus k})}{P(x_k^{(t)}|\mathbf{x}'_{\setminus k})P(\mathbf{x}'_{\setminus k})P(x'_k|\mathbf{x}'_{\setminus k})} = 1 \end{aligned}$$

- The samples are always accepted

Example of Gibbs Sampling

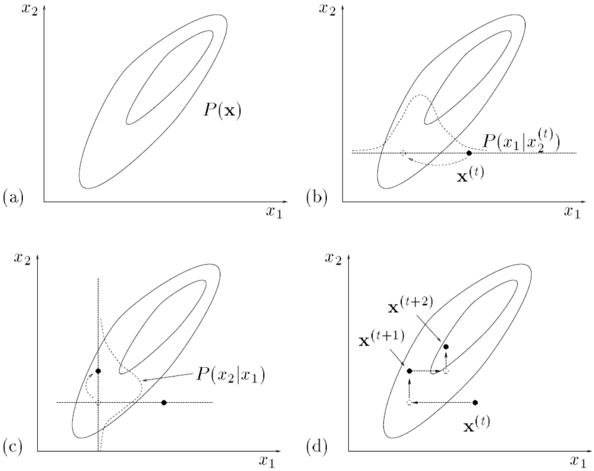


Figure 9. Gibbs sampling. (a) The joint density $P(\mathbf{x})$ from which samples are required. (b) Starting from a state $\mathbf{x}^{(t)}$, x_1 is sampled from the conditional density $P(x_1|x_2^{(t)})$. (c) A sample is then made from the conditional density $P(x_2|x_1)$. (d) A couple of iterations of Gibbs sampling.

References I

MacKay, D. J. C. (1998). Introduction to Monte Carlo methods. In Jordan, M. I., editor, *Learning in Graphical Models*, NATO Science Series, pages 175–204. Kluwer Academic Press.