# LDA Essentials

Yong Sun
Lizhang Zhan

# Mixture Topic Modeling
## — General Ideas

*Topics*

| | |
|---|---|
| gene | 0.04 |
| dna | 0.02 |
| genetic | 0.01 |
| . . . | |

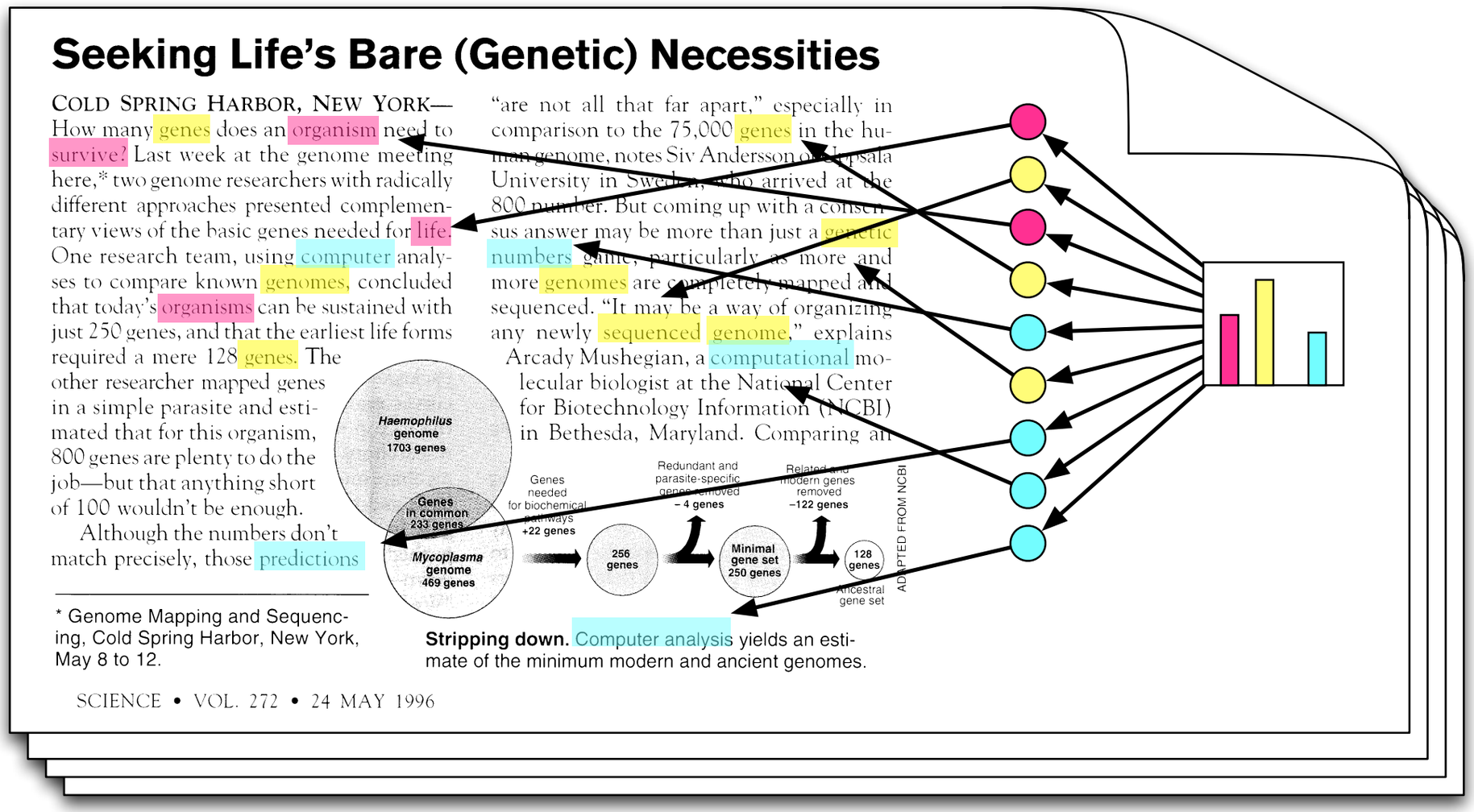
| | |
|---|---|
| life | 0.02 |
| evolve | 0.01 |
| organism | 0.01 |
| . . . | |

| | |
|---|---|
| brain | 0.04 |
| neuron | 0.02 |
| nerve | 0.01 |
| . . . | |

| | |
|---|---|
| data | 0.02 |
| number | 0.02 |
| computer | 0.01 |
| . . . | |

*Documents*

*Topic proportions and assignments*



**Seeking Life's Bare (Genetic) Necessities**

COLD SPRING HARBOR, NEW YORK— How many genes does an organism need to survive? Last week at the genome meeting here,* two genome researchers with radically different approaches presented complementary views of the basic genes needed for life. One research team, using computer analyses to compare known genomes, concluded that today's organisms can be sustained with just 250 genes, and that the earliest life forms required a mere 128 genes. The other researcher mapped genes in a simple parasite and estimated that for this organism, 800 genes are plenty to do the job—but that anything short of 100 wouldn't be enough.

Although the numbers don't match precisely, those predictions "are not all that far apart," especially in comparison to the 75,000 genes in the human genome, notes Siv Andersson of Uppsala University in Sweden, who arrived at the 800 number. But coming up with a consensus answer may be more than just a genetic numbers game, particularly as more and more genomes are completely mapped and sequenced. "It may be a way of organizing any newly sequenced genome," explains Arcady Mushegian, a computational molecular biologist at the National Center for Biotechnology Information (NCBI) in Bethesda, Maryland. Comparing an



* Genome Mapping and Sequencing, Cold Spring Harbor, New York, May 8 to 12.

**Stripping down.** Computer analysis yields an estimate of the minimum modern and ancient genomes.

SCIENCE • VOL. 272 • 24 MAY 1996

# The Generative Story

- Each **topic** is a distribution over words
- Each **document** is a mixture of corpus-wide topics (one document may contain several aspects, i.e., topics)
- Each **word** is drawn from one of these topics

# Basic Probability Rules

$$p(\vartheta|\mathcal{X}) = \frac{p(\mathcal{X}|\vartheta) \cdot p(\vartheta)}{p(\mathcal{X})}, \qquad \text{posterior} = \frac{\text{likelihood} \cdot \text{prior}}{\text{evidence}}.$$

```
p(x, y) = p(x)•p(y|x)
        = p(y)•p(x|y)
```

Marginal distribution:

```
p(x) = ∑x p(x, y)     :discrete scenario
     = ∫p(x,y)dy      :continuous scenario
```

Expectation:

```
Exp(X) = ∑x p(x)•x
       = ∫p(x)•x dx
```

# Sampling from a Distribution

or draw a sample from a given distribution

- rand(3C) could generate pseudo random numbers uniformly, or in another word draw samples from a uniform distribution.
- How to sample from a non-uniform distribution? E.g., a multinomial distribution $P(X) = \{X_1:20\%, X_2:40\%, X_3:30\%, X_4:10\%\}$

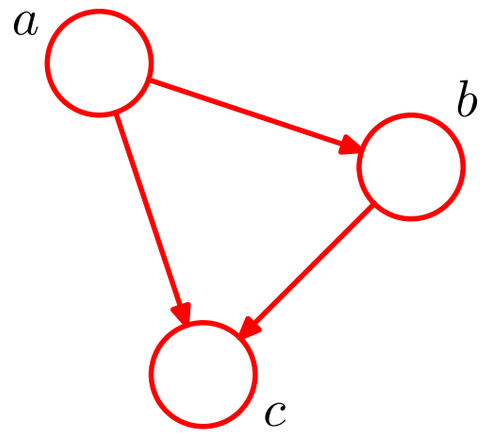
1) divide the range [0, 1.0] to 4 parts:
   [0, 0.4), [0.4, 0.7), [0.7, 0.9), [0,9, 1.0]

2) generate a random number in [0, 1.0], if it falls into the 2nd range, choose and return $X_3$ ...

```
>>> numpy.random.multinomial (1, (0.2, 0.4, 0.4, 0.1))
array([0, 1, 0, 0])
```

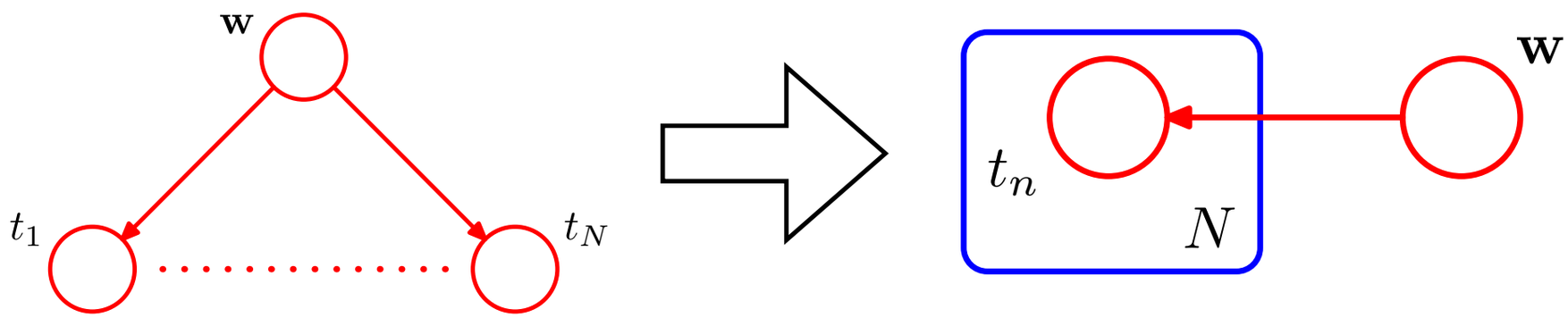
# Graphical Model

## — DAG (Directed Acyclic Graph)




$$p(a, b, c) = p(c|a, b)p(b|a)p(a).$$

plate notation:




$$p(\mathbf{t}, \mathbf{w}) = p(\mathbf{w}) \prod_{n=1}^{N} p(t_n|\mathbf{w}).$$

# Dirichlet Distribution

- A distribution of distributions.
- governed by a hyper-parameter, vector $\alpha = \{\alpha_1, \alpha_2, \alpha_3 \ldots \alpha_K\}$
- each sample from Dir($\alpha$) is a multinomial distribution (K-dimension).

```
>>> numpy.random.dirichlet((0.5,)*6, 1)
array([[  2.16869082e-01,   4.84643911e-01,   3.05685740e-02,
          5.72632328e-02,   2.10586822e-01,   6.83778288e-05]])
```

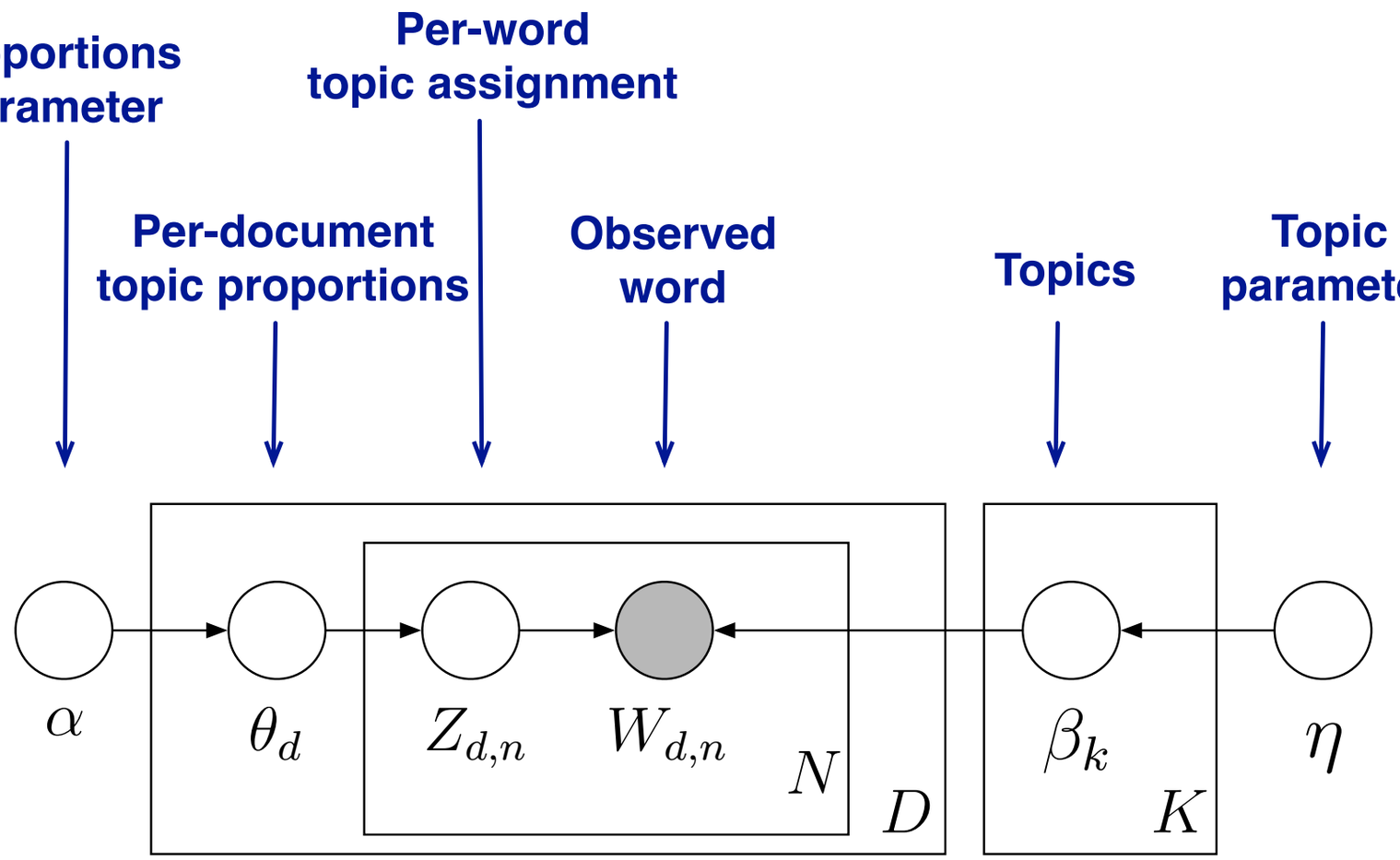
- Conjugate to multinomial distribution.
- http://en.wikipedia.org/wiki/Dirichlet_distribution

# LDA (Latent Dirichlet Allocation)

Latent Dirichlet Allocation (LDA) is a generative probabilistic model of a corpus. The basic idea is documents are represented as random mixtures over latent topics, where each topic is characterized by a distribution over words.

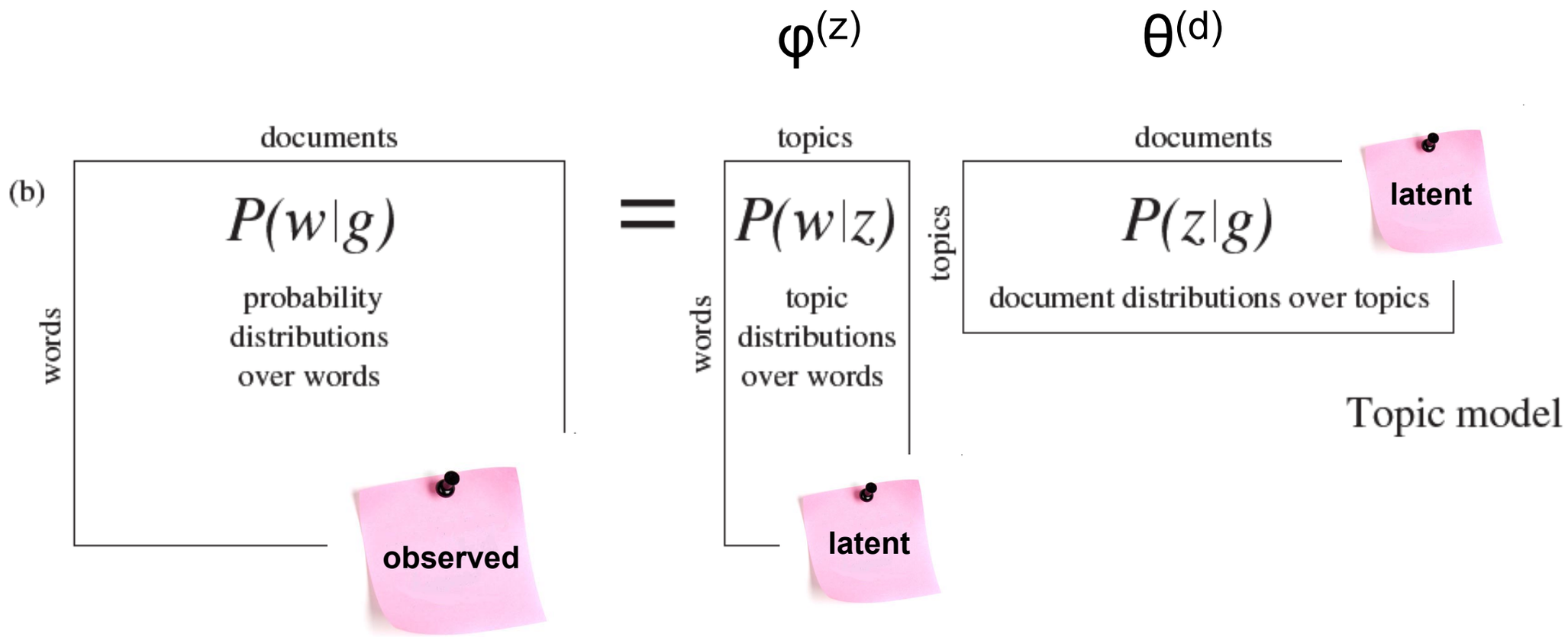
1. Choose $N \sim \text{Poisson}(\xi)$.
2. Choose $\theta \sim \text{Dir}(\alpha)$.
3. For each of the $N$ words $w_n$:
   (a) Choose a topic $z_n \sim \text{Multinomial}(\theta)$.
   (b) Choose a word $w_n$ from $p(w_n \mid z_n, \beta)$, a multinomial probability conditioned on the topic $z_n$.

# LDA as a Graphical Model




$$\prod_{i=1}^{K} p(\beta_i \mid \eta) \prod_{d=1}^{D} p(\theta_d \mid \alpha) \left( \prod_{n=1}^{N} p(z_{d,n} \mid \theta_d) p(w_{d,n} \mid \beta_{1:K}, z_{d,n}) \right)$$

# Matrix Representation of LDA



# Parameter Estimation

Approximate posterior inference algorithms

- Mean field variational methods (Blei 2001, 2003)
- Expectation propagation (Minka and Lafferty, 2002)
- Collapsed Gibbs Sampling (Griffiths and Steyvers, 2002)
- Collapsed variational inference (Teh et al., 2006)
- Online variational inference (Hoffman et al., 2010)

# MCMC and Gibbs Sampling

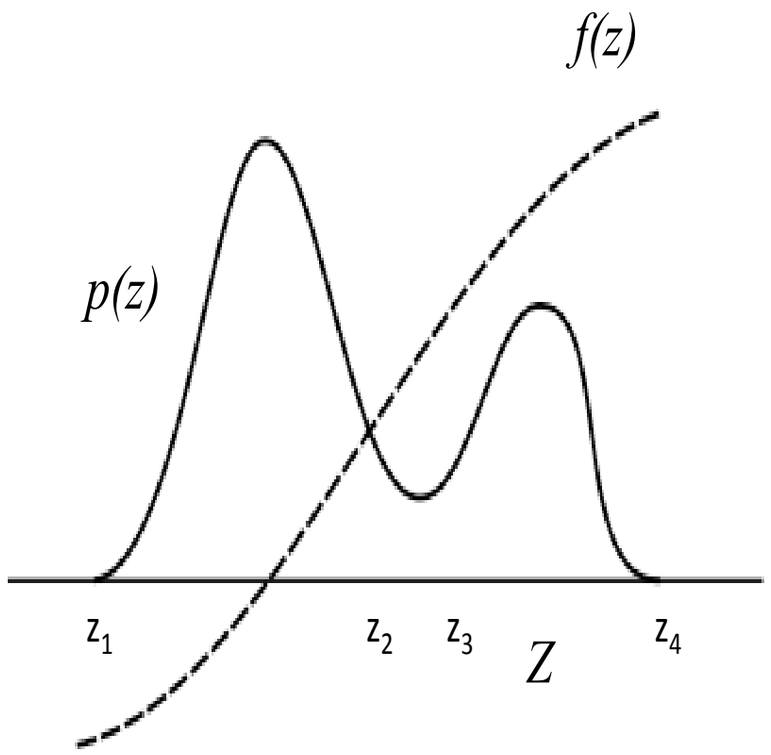
## Monte Carlo Integration:

$$\int_a^b h(x)\,dx = \int_a^b f(x)\,p(x)\,dx = E_{p(x)}[\,f(x)\,]$$

If we can decompose h(x) into the production of function f(x) and probability density function p(x) defined over the interval (a, b). So that the integral can be expressed as an expectation of f(x) over p(x). Thus if we draw a large number $x_i$,

$$\int_a^b h(x)\,dx = E_{p(x)}[\,f(x)\,] \simeq \frac{1}{n}\sum_{i=1}^{n} f(x_i)$$

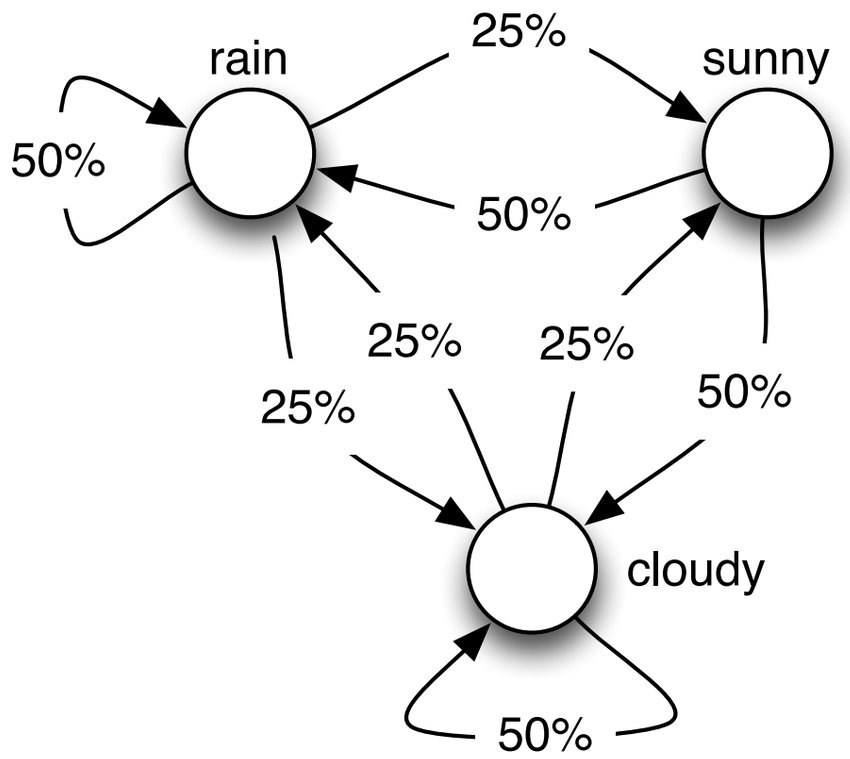
# Sampling for MC Integration




To calculate the integral more effectively and more accurately, we should sample as much as possible from $[Z_1, Z_2]$ and $[Z_3, Z_4]$.

# Markov Chain

$$\Pr(X_{t+1} = s_j \,|\, X_0 = s_k, \cdots, X_t = s_i) = \Pr(X_{t+1} = s_j \,|\, X_t = s_i)$$




$$\pi_j(t) = \Pr(X_t = s_j)$$

$$\boldsymbol{\pi}(t+1) = \boldsymbol{\pi}(t)\mathbf{P}$$

$$\mathbf{P} = \begin{pmatrix} 0.5 & 0.25 & 0.25 \\ 0.5 & 0 & 0.5 \\ 0.25 & 0.25 & 0.5 \end{pmatrix}$$

$$\boldsymbol{\pi}(0) = \begin{pmatrix} 0 & 1 & 0 \end{pmatrix}$$

$$\boldsymbol{\pi}(7) = \boldsymbol{\pi}(0)\mathbf{P}^7 = \begin{pmatrix} 0.4 & 0.2 & 0.4 \end{pmatrix}$$

```
>>> import numpy as np
>>> p = np.matrix(((0.5, 0.25, 0.25), (0.5, 0, 0.5), (0.25, 0.25, 0.5)))
>>> print (0, 1, 0) * (p ** 7)
[[ 0.40002441  0.19995117  0.40002441]]
```

# MCMC

- MCMC (Markov Chain Monte Carlo)
  - z^0 is initialized randomly (or uniformly)
  - having a sample z^t on time t
  - draw a new sample z^(t+1) from P_trans(z|z^t)
- Metropolis-Hastings algorithm is proven to construct a Markov Chain which is irreducible, aperiodic and convergent.
- Gibbs Sampling is a special case of M-H algorithm, and require z has at least 2 dimensions..

# Gibbs Sampler

1. Initialize $\{z_i : i = 1, \ldots, M\}$
2. For $\tau = 1, \ldots, T$:
   - Sample $z_1^{(\tau+1)} \sim p(z_1 | z_2^{(\tau)}, z_3^{(\tau)}, \ldots, z_M^{(\tau)})$.
   - Sample $z_2^{(\tau+1)} \sim p(z_2 | z_1^{(\tau+1)}, z_3^{(\tau)}, \ldots, z_M^{(\tau)})$.
   - $\vdots$
   - Sample $z_j^{(\tau+1)} \sim p(z_j | z_1^{(\tau+1)}, \ldots, z_{j-1}^{(\tau+1)}, z_{j+1}^{(\tau)}, \ldots, z_M^{(\tau)})$.
   - $\vdots$
   - Sample $z_M^{(\tau+1)} \sim p(z_M | z_1^{(\tau+1)}, z_2^{(\tau+1)}, \ldots, z_{M-1}^{(\tau+1)})$.

Note: Gibbs Sampling is difficult to handle non-conjugacy; it is hard to generalize to the dynamic topic model and correlated topic model.

# Gibbs Sampling for LDA

- Represent corpus as an array of words w[i], document indices d[i] and topics z[i], only topics z[i] change.

- States of Markov Chain: topic assignments to words.

Define $n_{-i,j}(w_i)$ as freq. of $w_i$ labeled as topic $j$.
Define $n_{-i,j}(d_i)$ as number of words in $d_i$ labeled as topic $j$.

$$P(z_i = j|\mathbf{z}_{-i}, \mathbf{w}, \mathbf{d}) \quad \propto \quad \underbrace{\frac{n_{-i,j}(w_i) + \eta}{V\eta + \sum_{v=1}^{V} n_{-i,j}(w_v)}}_{\text{Prob of } w_i \text{ under topic } z_i} \qquad \underbrace{\frac{n_{-i,j}(d_i) + \alpha}{K\alpha \sum_{k=1}^{K} n_{-i,j}(d_i)}}_{\text{Prob of topic } z_i \text{ in document } d_i}$$

# Experiments on Solaris *man1**

- GibbsLDA++ (v0.2) & ompi-lda
- Recommended parameters: alpha = 50 / K, beta = 0.01 (K: topic numbers)
- Experiment data and results: /net/vigor.cn/projects/yongsun/refman

# Demo

```
$ GibbsLDA++-0.2/src/lda -est -alpha 1.0 -beta 0.01 -ntopics 50 -niters 10000 -savestep 500
  -twords 20 -dfile data/trndocs.dat

$ cat data/model-final.twords
```

```
Topic 39th:
        address   0.044344
        interface   0.041938
        ip   0.025722
        command   0.023075
        packet   0.022593
        link   0.021661
        system   0.020608
        administration   0.016877
        sunos   0.015614
        change   0.015163
        network   0.015103
        property   0.011252
        field   0.010078
        dladm   0.009447
        route   0.009296
        jul   0.008634
        protocol   0.008574
        policy   0.008213
        entry   0.008153
        routing   0.007431
Topic 49th:
        system   0.058070
        property   0.050747
        file   0.035296
        zfs   0.034587
        command   0.028539
        mount   0.027311
        snapshot   0.019751
        pool   0.019751
        nfs   0.018806
        share   0.016018
        administration   0.015971
        change   0.014459
        sunos   0.012994
        default   0.012758
        dataset   0.012285
        mounted   0.011293
        type   0.011104
        filesystem   0.011104
        set   0.010679
        option   0.010254
Topic 37th:
        locale   0.032980
        encoding   0.029275
        map   0.027968
        code   0.024699
        option   0.022447
        module   0.022447
        character   0.020776
        file   0.019251
        change   0.018742
        mapping   0.017653
        name   0.014965
        data   0.013948
        convert   0.013803
        icu   0.013439
        generate   0.012568
        user   0.011623
        environment   0.011478
        language   0.010752
        generated   0.009953
        ff   0.009807
```

```
$ ./topdocs.py | less

TOPIC 39:
  0.741350: ipadm.1m
  0.726053: ifconfig.1m
  0.632332: dladm.1m
  0.629213: flowadm.1m
  0.622549: ipmpstat.1m
  0.579861: if_mpadm.1m
  0.574853: snoop.1m
  0.574528: ipsecconf.1m
  0.573074: route.1m
  0.570520: traceroute.1m
  0.554011: in.mpathd.1m
  0.518006: ping.1m
  0.506318: in.routed.1m
  0.499579: routed.1m
  0.489180: netstat.1m
  0.454013: ilbadm.1m
  0.446224: arp.1m
  0.443049: ipseckey.1m
  0.440882: in.ndpd.1m
  0.431858: ipaddrsel.1m
  0.415822: in.ripngd.1m
  0.380814: 6to4relay.1m
  0.374824: tcpdump.1
  0.336815: rdisc.1m
  0.334204: in.rdisc.1m
  0.328594: ipf.1m
  0.327247: dhcpagent.1m
  0.326886: ipfstat.1m
  0.321940: ipsecalgs.1m
  0.316121: routeadm.1m
  0.288889: ipmon.1m
  0.285068: wpad.1m
  0.269076: ipnat.1m
  0.268012: sppptun.1m
  0.264261: svc.ipfd.1m
  0.254766: snmpnetstat.1
  0.248193: dhcpinfo.1
  0.230444: pppstats.1m
  0.217712: rtquery.1m
  0.214978: mibiisa.1m
  0.209472: flowstat.1m
  0.200238: dlstat.1m
  0.198502: ifparse.1m

TOPIC 49:
  0.759857: zfs_allow.1m
  0.742297: vdiskadm.1m
  0.732419: zfs_share.1m
  0.699644: zfs.1m
  0.594985: zfs_encrypt.1m
  0.577713: share.1m
  0.514714: zpool.1m
  0.478947: umount.1m
  0.459211: mount.1m
  0.455930: mount_smbfs.1m
  0.449402: umount_smbfs.1m
  0.406699: quotaon.1m
  0.398143: mount_nfs.1m
  0.384259: dfmounts.1m
  0.377990: quotaoff.1m
  0.374718: lockfs.1m
  0.370861: sharectl.1m
  0.363261: share_nfs.1m
  0.360000: unshare_nfs.1m
  0.353846: shareall.1m
  0.341317: unshare.1m
  0.333714: share_smb.1m
  0.330000: dfshares_nfs.1m
  0.327381: dfmounts_nfs.1m
  0.326316: quota.1m
  0.315385: unshareall.1m
  0.304813: dfshares.1m
  0.279433: automount.1m
  0.268293: mountd.1m
  0.260917: beadm.1m
  0.255499: nfsd.1m
  0.249284: ippool.1m
  0.237037: exportfs.1b
  0.234818: mount_ufs.1m
  0.232558: automountd.1m
  0.230088: quotacheck.1m
  0.223776: edquota.1m
  0.223377: lockd.1m
  0.216783: repquota.1m
  0.206897: showmount.1m
  0.183117: fssnap_ufs.1m
  0.182382: nfsstat.1m
  0.163717: rmumount.1

TOPIC 37:
  0.598775: uconv.1
  0.558611: idnconv.1
  0.464020: strconf.1
  0.449132: strchg.1
  0.436975: makeconv.1
  0.417465: idmap.1m
  0.413174: gencnval.1
  0.405128: iconv.1
  0.402985: genctd.1
  0.400602: derb.1
  0.389706: genrb.1
  0.375661: gensprep.1m
  0.374459: rsautl.1openssl
  0.353846: dumpcs.1
  0.350622: localectr.1m
  0.346154: genbrk.1
  0.332518: geniconvtbl.1
  0.330357: gencmn.1m
  0.329571: fsexam.1
  0.309963: auto_ef.1
  0.303614: pkgdata.1
  0.299342: locale.1
  0.293233: brltty.1
  0.292758: localedef.1
  0.287037: convmv.1
  0.281664: setterm.1
  0.263682: genccode.1m
  0.261261: encoding.1t
  0.260116: uxterm.1
  0.246479: icu-config.1
  0.241458: glib-genmarshal.1
  0.240678: espeak.1
  0.237931: flex.1
  0.237288: bdftruncate.1
  0.229777: msgcat.1t
  0.228571: regcmp.1
  0.222930: espeak-synthesis-driver.1
  0.221865: trietool-0.2.1
  0.217765: icupkg.1m
  0.213873: koi8rxterm.1
  0.213439: idn.1
  0.206030: uudecode.1c
  0.205128: scanimage.1
```

# Future Work

- phrase-based LDA: ‘file system’ makes more sense than ‘file’ and ‘system’, or adopt bi-gram in LDA modeling.
- Extract only the terms/phrases from the synopsis and description sections in man-pages, by leveraging the nroff parser being developed by Kaz.
- Considering other properties in man-pages, like see-also, package origin etc.
  - Investigate other topic models (LDA extensions) which is capable for meta information.
  - Or use the LDA result as one feature, for future clustering.

# References

- https://stbeehive.oracle.com/teamcollab/wiki/Systems+Globalization+Information+Engineering:LDA+and+Topic+Models

# Q & A