CPSC 440/540: Advanced Machine Learning

HMMs and Topic Models

Danica Sutherland (building on materials from Mark Schmidt)

University of British Columbia

Winter 2023

Quick things

admin

- I'm working on project proposal feedback... hopefully by tomorrow
- UBC participating in ASA Data Fest for the first time this year
  - Undergraduate data science hackathon, April 28th (5pm) to April 30th (6pm)
  - Register by April 10th: `https://ubc.ca1.qualtrics.com/jfe/form/SV_8ABL52tzvw2Z3rU`
  - Grad students can help as mentors – contact Giulia Toti (`gtoti@cs.ubc.ca`)

Last Time: Expectation Maximization

- EM considers learning with observed data $\mathbf{X}$ and hidden data $\mathbf{Z}$.
- What we'd really like to do is maximize the marginal log-likelihood:

$$\Theta \in \underset{\Theta}{\arg\max} \log \int_{\mathbf{Z}} p(\mathbf{X}, \mathbf{Z} \mid \Theta)\, d\mathbf{Z}$$

- EM is helpful when "complete" likelihood, $p(\mathbf{X}, \mathbf{Z} \mid \Theta)$, has a nice form.
- EM iterations take the form of a weighted "complete" MLE,

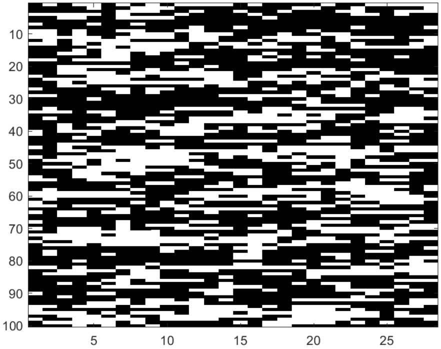
$$\Theta^{t+1} \in \underset{\Theta}{\arg\max} \int_{\mathbf{Z}} p(\mathbf{Z} \mid \mathbf{X}, \Theta^t) \log p(\mathbf{X}, \mathbf{Z} \mid \Theta)\, d\mathbf{Z},$$

  taking an expectation over $\mathbf{Z}$ w.r.t. the *previous* Θ^t.
- We looked at the simple form of the EM update for mixture models,

$$\Theta^{t+1} \in \underset{\Theta}{\arg\max} \sum_{i=1}^{n} \sum_{z^i=1}^{k} \underbrace{p(z^i \mid x^i, \Theta^t)}_{\text{responsibility}} \underbrace{\log p(x^i, z^i \mid \Theta)}_{\text{complete-data log-lik}}.$$

Back to the Rain Data

- We previously considered the "Vancouver Rain" data:



- We used homogeneous Markov chains to model between-day dependence.

Back to the Rain Data

- Before, we used a conditional random field to depend on the month.
- We could alternately try to learn the clusters using a mixture model.
  - But mixture of independents wouldn't capture dependencies within cluster.
- A mixture of Markov chains could capture direct dependence *and* clusters,

$$p(x_1, x_2, \ldots, x_d) = \sum_{c=1}^{k} p(z = c) \underbrace{p(x_1 \mid z = c)p(x_2 \mid x_1, z = c) \cdots p(x_d \mid x_{d-1}, z = c)}_{\text{Markov chain for cluster } c}.$$

- Cluster z chooses which homogeneous Markov chain parameters to use.
  - We could learn that some months are more likely to have rain (like winter months).
  - Can do inference by running forward-backward on each mixture; fit model with EM.

Comparison of Models on Rain Data

- Independent (homogeneous) Bernoulli:
  - Average NLL: 18.97 (1 parameter).
- Independent Bernoullis:
  - Average NLL: 18.95, (28 parmaeters).
- Mixture of Bernoullis ($k = 10$, five random restarts of EM):
  - Average NLL: 17.06 ($10 + 10 \times 28 = 290$ parameters)
- Homogeneous Markov chain:
  - Average NLL: 16.81 (3 parameters)
- Mixture of Markov chains ($k = 10$, five random restarts of EM):
  - Average NLL: 16.53 ($10 + 10 \times 3 = 40$ parameters).
  - Parameters of one of the clusters (possibly modeling summer months):

$$
\begin{aligned}
&p(z = 5) = 0.14 \\
&p(x_1 = \text{“rain”} \mid z = 5) = 0.22 && \text{(instead of usual 37\%)} \\
&p(x_j = \text{“rain”} \mid x_{j-1} = \text{“rain”}, z = 5) = 0.49 && \text{(instead of usual 65\%)} \\
&p(x_j = \text{“rain”} \mid x_{j-1} = \text{“not rain”}, z = 5) = 0.11 && \text{(instead of usual 35\%)}
\end{aligned}
$$

Back to the Rain Data

- The rain data is artificially divided into months.

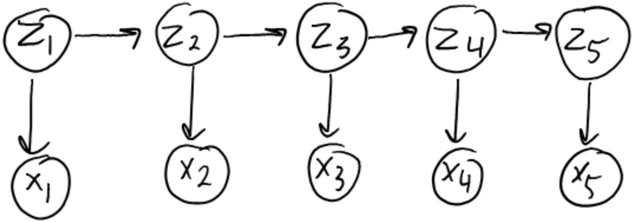
- We previously discussed viewing rain data as one very long sequence ($n = 1$).
- We could apply homogeneous Markov chains due to parameter tying.
  - But a mixture doesn't make sense when $n = 1$.

- What we want: different "parts" of the sequence come from different clusters.
  - We transition from "summer" cluster to "fall" cluster at some time j.

- One way to address this is with a "hidden" Markov model (HMM):
  - Instead of examples being assigned to clusters, days are assigned to clusters.
  - Have a Markov dependency between cluster values of adjacent days.

Hidden Markov Models

- Hidden Markov models have each x_j depend on a hidden Markov chain.

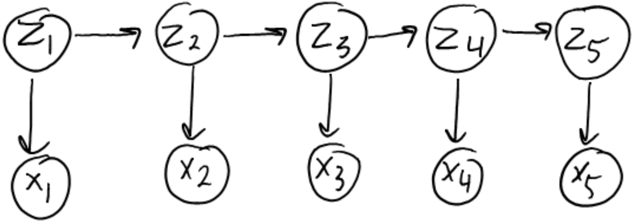


$$p(x_1, x_2, \ldots, x_d, z_1, z_2, \ldots z_d) = p(z_1) \prod_{j=2}^{d} p(z_j \mid z_{j-1}) \prod_{j=1}^{d} p(x_j \mid z_j).$$

- We're going to learn clusters z_j and the hidden dynamics between days.
  - Hidden cluster z_j could be "summer" or "winter" (we're learning the clusters).
  - Transition probability $p(z_j \mid z_{j-1})$ is probability of staying in "summer".
    - Initial probability $p(z_1)$ is probability of starting chain in "summer".
  - Emission probability $p(x_j \mid z_j)$ is probability of `rain` during "summer".

Hidden Markov Models

- Hidden Markov models have each x_j depend on a hidden Markov chain.



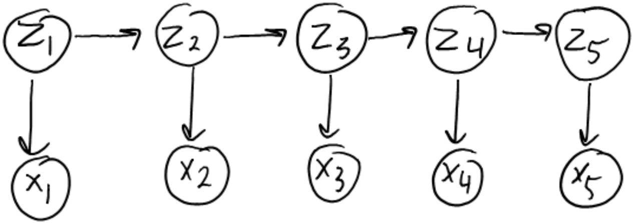
$$p(x_1, x_2, \ldots, x_d, z_1, z_2, \ldots z_d) = p(z_1) \prod_{j=2}^{d} p(z_j \mid z_{j-1}) \prod_{j=1}^{d} p(x_j \mid z_j).$$

- You observe the x_j values but don't see the z_j values.
  - There is a "hidden" Markov chain, whose state determines the cluster at each time.

- HMMs generalize both Markov chains and mixture of categoricals.
  - Both models are obtained under appropriate parameters.

Hidden Markov Models

- Hidden Markov models have each x_j depend on a hidden Markov chain.



$$p(x_1, x_2, \ldots, x_d, z_1, z_2, \ldots z_d) = p(z_1) \prod_{j=2}^{d} p(z_j \mid z_{j-1}) \prod_{j=1}^{d} p(x_j \mid z_j).$$

- Note that the x_j can be continuous even with discrete clusters z_j.
  - Data could come from a mixture of Gaussians, with cluster changing in time.
- If the z_j are continuous it's often called a state-space model.
  - If everything is Gaussian, it leads to Kalman filtering.
  - Keywords for non-Gaussian: unscented Kalman filter and particle filter.

Applications of HMMs and Kalman Filters

- HMMs variants are probably the most-used time-series model.

Applications [edit]

HMMs can be applied in many fields where the goal is to recover a data sequence that is not immediately observable (but other data that depend on the sequence are). Applications include:

- Single Molecule Kinetic analysis[16]
- Cryptanalysis
- Speech recognition
- Speech synthesis
- Part-of-speech tagging
- Document Separation in scanning solutions
- Machine translation
- Partial discharge
- Gene prediction
- Alignment of bio-sequences
- Time Series Analysis
- Activity recognition
- Protein folding[17]
- Metamorphic Virus Detection[18]
- DNA Motif Discovery[19]

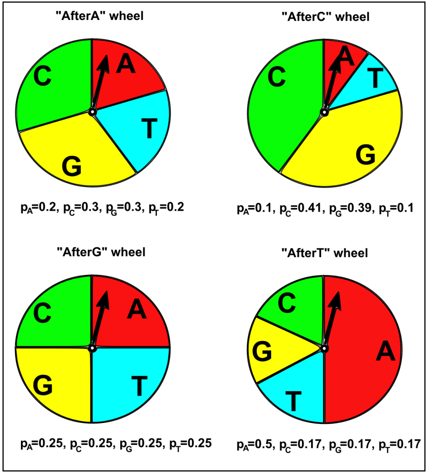
Applications [edit]

- Attitude and Heading Reference Systems
- Autopilot
- Battery state of charge (SoC) estimation[39][40]
- Brain-computer interface
- Chaotic signals
- Tracking and Vertex Fitting of charged particles in Particle Detectors[41]
- Tracking of objects in computer vision
- Dynamic positioning
- Economics, in particular macroeconomics, time series analysis, and econometrics[42]
- Inertial guidance system
- Orbit Determination
- Power system state estimation
- Radar tracker
- Satellite navigation systems
- Seismology[43]
- Sensorless control of AC motor variable-frequency drives
- Simultaneous localization and mapping
- Speech enhancement
- Visual odometry
- Weather forecasting
- Navigation system
- 3D modeling
- Structural health monitoring
- Human sensorimotor processing[44]

Also includes chain-structured conditional random fields.

Example: Modeling DNA Sequences

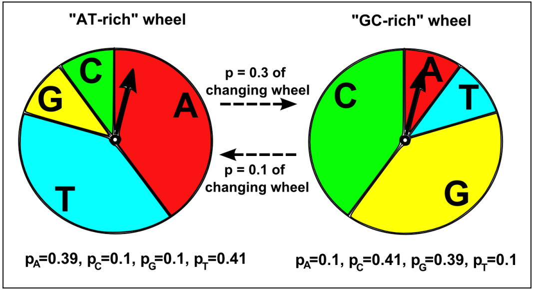
- Previously: Markov chain for DNA sequences:



https://www.tes.com/lessons/WE5E9RncBhieAQ/dna

Example: Modeling DNA Sequences

- Hidden Markov model (HMM) for DNA sequences (two hidden clusters):



- This is a (hidden) state transition diagram.
  - Can reflect that probabilities are different in different regions.
  - The actual regions are not given, but instead are nuissance variables handled by EM.

- A better model might use a hidden and visible Markov chain.
  - With 2 hidden clusters, you would have 8 “probability wheels” (4 per cluster).
  - Would have “treewidth 2”, so inference would be tractable.

Inference and Learning in HMMs

- Given observed features x_j, likelihood of a joint z_j assignment is

$$p(z_1, z_2, \dots z_d \mid x_1, x_2, \dots, x_d) \propto p(z_1) \prod_{j=2}^{d} p(z_j \mid z_{j-1}) \prod_{j=1}^{d} p(x_j \mid z_j).$$

- We can do inference with forward-backward by converting to potentials:

$$\begin{aligned} \phi_1(z_1) &= p(z_1)p(x_1 \mid z_1) \\ \phi_j(z_j) &= p(x_j \mid z_j) && (j > 1) \\ \phi_{j,j-1}(z_j, z_{j-1}) &= p(z_j \mid z_{j-1}). \end{aligned}$$

- Marginals from forward-backward are used to update parameters in EM.
  - In this setting EM is called the "Baum-Welch" algorithm.
  - As with other mixture models, learning with EM is sensitive to initialization.

Who is Guarding Who?

bonus!

- There is a lot of data on scoring/offense of NBA basketball players.
  - Every point and assist is recorded, more scoring gives more wins and $$$.

- But how do we measure defense ("stopping people from scoring")?
  - We need to know who each player is guarding (which is not recorded)

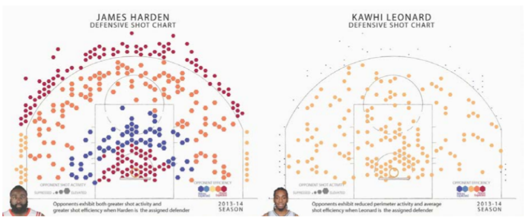


Figure 2a. Graphical depiction of a defender's volume (size) and disruption scores (color). Kawhi Leonard tends to suppress shots on the perimeter. More comparisons are provided in the Appendix.

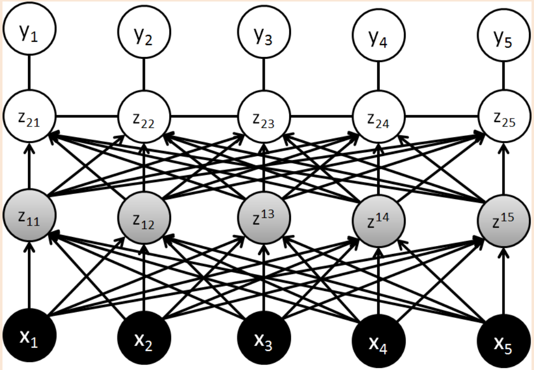
http://www.lukebornn.com/papers/franks_ssac_2015.pdf

- HMMs can be used to model who is guarding who over time.
  - https://www.youtube.com/watch?v=JvNkZdZJBt4

Neural Networks with Latent-Dynamics

bonus!

- Could have (undirected) HMM parameters come out of a neural network:
  - Tries to model hidden dynamics across time.



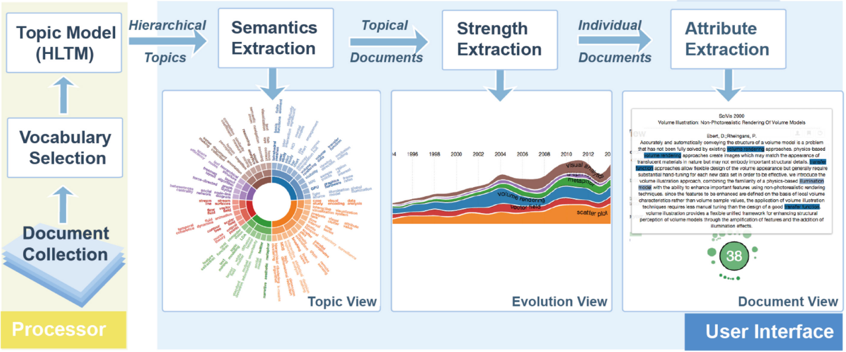
- Combines deep learning, mixture models, and graphical models.
  - "Latent-dynamics model".
  - Previously achieved among state of the art in several applications.

Outline

Motivation for Topic Models

We want a model of the hidden "factors" making up a set of documents.

- In this context, latent-factor models are called topic models.



https://www.sciencedirect.com/science/article/pii/S2468502X17300074

- "Topics" could be useful for things like searching for relevant documents.

Classic Approach: Latent Semantic Indexing

- Classic methods are based on scores like TF-IDF:
  1. Term frequency: probability of a word occuring within a document.
     - E.g., 7% of words in document i are `the` and 2% of the words are `LeBron`.
  2. Document frequency: probability of a word occuring across documents.
     - E.g., 100% of documents contain `the` and 0.01% have `LeBron`.
  3. TF-IDF: measures like (term frequency)*log 1/(document frequency).
     - Seeing `LeBron` tells you a lot about the document; seeing `the` tells you nothing.

- Many many many variations exist.

- TF-IDF features are very redundant.
  - Consider TF-IDF of `LeBron`, `Durant`, and `Giannis`.
  - High values of these typically just indicate topic of "basketball".
  - Basically a weighted bag of words.

- We want to find latent factors ("topics") like "basketball".

Modern Approach: Latent Dirichlet Allocation

- Latent semantic indexing (LSI) topic model:
  1. Summarize each document by its TF-IDF values.
  2. Run a latent-factor model like PCA or NMF on the matrix.
  3. Treat the latent factors as the “topics”.

- LSI has been largely replaced by latent Dirichlet allocation (LDA).
  - Hierarchical Bayesian model of all words in a document.
    - Still ignores word order.
    - Tries to explain all words in terms of topics.
  - The most cited ML paper in the 00s?

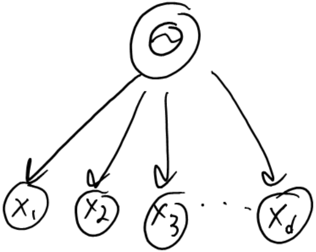
- LDA has several components; we’ll build up to it by parts.
  - We’ll assume all documents have d words and word order doesn’t matter.

Model 1: Categorical Distribution of Words

- Base model: each word x_j comes from the same categorical distribution.

$$p(x_j = \texttt{the}) = \theta_{\texttt{the}} \quad \text{where} \quad \theta_{\text{word}} \geq 0 \quad \text{and} \quad \sum_{\text{word}} \theta_{\text{word}} = 1.$$

- So to generate a document with d words:
  - Sample d words from the categorical distribution.

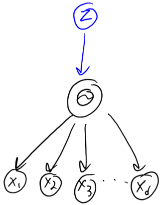


- Drawback: misses that documents are about different "topics."
  - We want the word distribution to depend on the "topics."

Model 2: Mixture of Categorical Distributions

- To represent "topics", we'll use a mixture model.
    - Each mixture has its own categorical distribution over words.
        - E.g., the "basketball" mixture will have higher probability of `LeBron`.

- So to generate a document with d words:
    - Sample a topic z from a categorical distribution.
    - Sample d words from categorical distribution z.

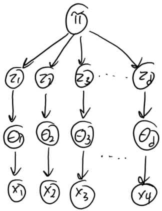


- Similar to a mixture of independent categorical distributions.
    - But we tie categorical distribution across the d variables, given cluster.
- Drawback: misses that documents may be about more than one topic.

Model 3: Multi-Topic Mixture of Categorical

- Our third model introduces a new vector of "topic proportions" π.
  - Gives percentage of each topic that makes up the document.
    - E.g., 80% basketball and 20% politics.
  - Called probabilistic latent semantic indexing (PLSI).

- So to generate a document with d words given topic proportions π:
  - Sample d topics z_j from categorical distribution π.
  - Sample a word for each z_j from corresponding categorical distribution.

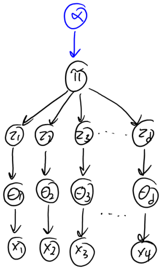


- Similar to HMM where each "time" has own cluster (but no Markov assumption).

Model 4: Latent Dirichlet Allocation

- Latent Dirichlet allocation (LDA) puts a prior on topic proportions.
  - Conjugate prior for categorical is Dirichlet distribution.

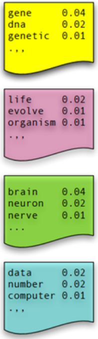
- So to generate a document with d words given Dirichlet prior:
  - Sample mixture proportions π from the Dirichlet prior.
  - Sample d topics z_j from categorical distribution π.
  - Sample a word for each z_j from corresponding categorical distribution.



- This is the generative model, typically used with MCMC or variational methods.

Latent Dirichlet Allocation (LDA)

Topics | *Documents* | *Topic proportions and assignments*



Each topic is like a "principal component" or "latent factor"

http://menome.com/wp/wp-content/uploads/2014/12/Blei2011.pdf

Latent Dirichlet Allocation (LDA)

1. Sample topic proportions Θ from Dirichlet.

Each topic is like a "principal component" or "latent factor"

http://menome.com/wp/wp-content/uploads/2014/12/Blei2011.pdf

Latent Dirichlet Allocation (LDA)

1. Sample topic proportions θ from Dirichlet.

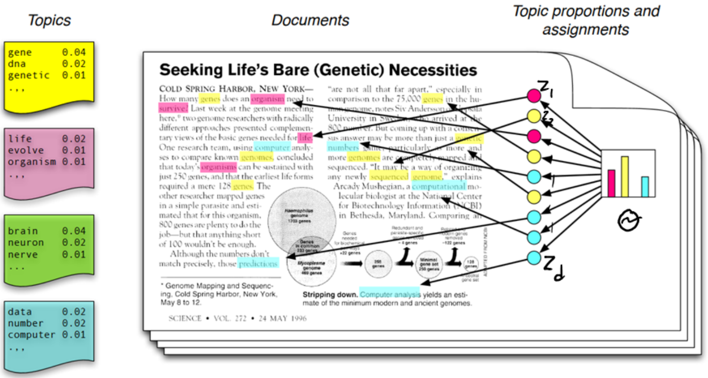
2. Sample 'd' topics z_j from θ.

→ Each topic is like a "principal component" or "latent factor"

http://menome.com/wp/wp-content/uploads/2014/12/Blei2011.pdf

Latent Dirichlet Allocation (LDA)

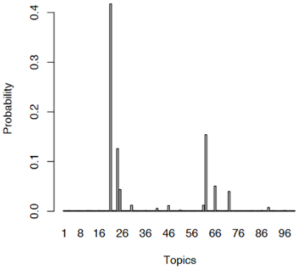
1. Sample topic proportions θ from Dirichlet.
2. Sample 'd' topics z_j from θ.
3. For each z_j sample a word based on frequencies for topic.



Each topic is like a "principal component" or "latent factor"

http://menome.com/wp/wp-content/uploads/2014/12/Blei2011.pdf

Latent Dirichlet Allocation Example



"Genetics"	"Evolution"	"Disease"	"Computers"
human	evolution	disease	computer
genome	evolutionary	host	models
dna	species	bacteria	information
genetic	organisms	diseases	data
genes	life	resistance	computers
sequence	origin	bacterial	system
gene	biology	new	network
molecular	groups	strains	systems
sequencing	phylogenetic	control	model
map	living	infectious	parallel
information	diversity	malaria	methods
genetics	group	parasite	networks
mapping	new	parasites	software
project	two	united	new
sequences	common	tuberculosis	simulations

Figure 2: **Real inference with LDA.** We fit a 100-topic LDA model to 17,000 articles from the journal *Science*. At left is the inferred topic proportions for the example article in Figure 1. At right are the top 15 most frequent words from the most frequent topics found in this article.

http://menome.com/wp/wp-content/uploads/2014/12/Blei2011.pdf

Latent Dirichlet Allocation Example

Figure 3: A topic model fit to the *Yale Law Journal*. Here there are twenty topics (the top eight are plotted). Each topic is illustrated with its top most frequent words. Each word's position along the x-axis denotes its specificity to the documents. For example "estate" in the first topic is more specific than "tax."

http://menome.com/wp/wp-content/uploads/2014/12/Blei2011.pdf

Latent Dirichlet Allocation Example

Health topics in social media:

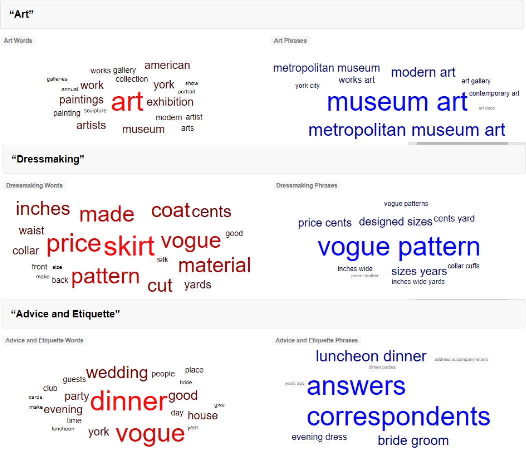
Non-Ailment Topics						
TV & Movies	**Games & Sports**	**School**	**Conversation**	**Family**	**Transportation**	**Music**
watch	killing	ugh	ill	mom	home	voice
watching	play	class	ok	shes	car	hear
tv	game	school	haha	dad	drive	feelin
killing	playing	read	ha	says	walk	lil
movie	win	test	fine	hes	bus	night
seen	boys	doing	yeah	sister	driving	bit
movies	games	finish	thanks	tell	trip	music
mr	fight	reading	hey	mum	ride	listening
watched	lost	teacher	thats	brother	leave	listen
hi	team	write	xd	thinks	house	sound

Ailments						
	Influenza-like Illness	**Insomnia & Sleep Issues**	**Diet & Exercise**	**Cancer & Serious Illness**	**Injuries & Pain**	**Dental Health**
General Words	better	night	body	cancer	hurts	dentist
	hope	bed	pounds	help	knee	appointment
	ill	body	gym	pray	ankle	doctors
	soon	ill	weight	awareness	hurt	tooth
	feel	tired	lost	diagnosed	neck	teeth
	feeling	work	workout	prayers	ouch	appt
	day	day	lose	died	leg	wisdom
	flu	hours	days	family	arm	eye
	thanks	asleep	legs	friend	fell	going
	xx	morning	week	shes	left	went
Symptoms	sick	sleep	sore	cancer	pain	infection
	sore	headache	throat	breast	sore	pain
	throat	fall	pain	lung	head	mouth
	fever	insomnia	aching	prostate	foot	ear
	cough	sleeping	stomach	sad	feet	sinus
Treatments	hospital	sleeping	exercise	surgery	massage	surgery
	surgery	pills	diet	hospital	brace	braces
	antibiotics	caffeine	dieting	treatment	physical	antibiotics
	fluids	pill	exercises	heart	therapy	eye
	paracetamol	tylenol	protein	transplant	crutches	hospital

http://journals.plos.org/plosone/article?id=10.1371/journal.pone.0103408

Latent Dirichlet Allocation Example

Three topics in 100 years of "Vogue" fashion magazine:



http://dh.library.yale.edu/projects/vogue/topics/

Discussion of Topic Models

- There are *many* extensions of LDA:
  - We can put prior on the number of words (like Poisson).
  - Correlated and hierarchical topic models learn dependencies between topics.

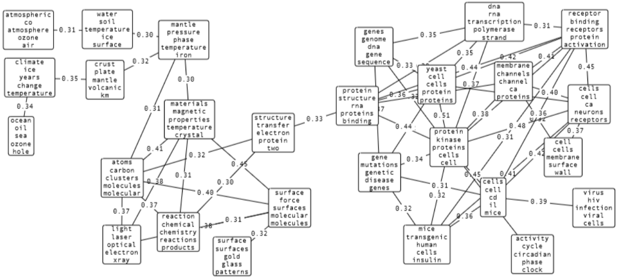
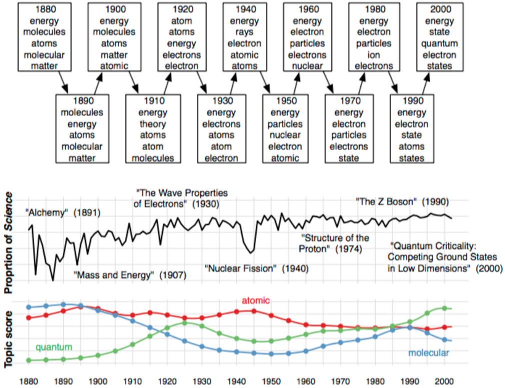


Figure 2: A portion of the topic graph learned from 15,744 OCR articles from *Science*. Each node represents a topic, and is labeled with the five most probable words from its distribution; edges are labeled with the correlation between topics.

http://people.ee.duke.edu/~lcarin/Blei2005CTM.pdf

Discussion of Topic Models

- There are *many* extensions of LDA:
  - We can put prior on the number of words (like Poisson).
  - Correlated and hierarchical topic models learn dependencies between topics.
  - Can be combined with Markov models to capture dependencies over time.



http://menome.com/wp/wp-content/uploads/2014/12/Blei2011.pdf

Discussion of Topic Models

- There are *many* extensions of LDA:
  - We can put prior on the number of words (like Poisson).
  - Correlated and hierarchical topic models learn dependencies between topics.
  - Can be combined with Markov models to capture dependencies over time.
  - Better word representations like "word2vec" (CPSC 340).
  - Now being applied beyond text, like "cancer mutation signatures":

http://journals.plos.org/plosgenetics/article?id=10.1371/journal.pgen.1005657

Discussion of Topic Models

- Topic models for analyzing musical keys:

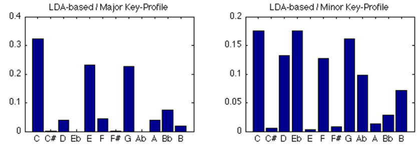


Figure 2: The C major and C minor key-profiles learned by our model, as encoded by the β matrix. Resulting key-profiles are obtained by transposition.

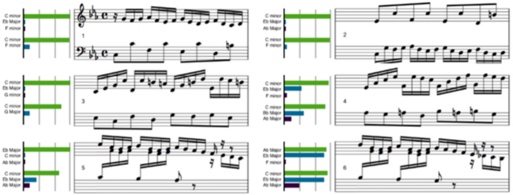


Figure 3: Key judgments for the first 6 measures of Bach's Prelude in C minor, WTC-II. Annotations for each measure show the top three keys (and relative strengths) chosen for each measure. The top set of three annotations are judgments from our LDA-based model; the bottom set of three are from human expert judgments [3].

http://cseweb.ucsd.edu/~dhu/docs/nips09_abstract.pdf

Monte Carlo Methods for Topic Models

- Nasty integrals in topic models:

Inference [edit]

See also: Dirichlet-multinomial distribution

Learning the various distributions (the set of topics, their associated word probabilities, the topic of each word, and the particular topic mixture of each document) is a problem of Bayesian inference. The original paper used a variational Bayes approximation of the posterior distribution;[1] alternative inference techniques use Gibbs sampling[6] and expectation propagation.[7]

Following is the derivation of the equations for collapsed Gibbs sampling, which means φs and θs will be integrated out. For simplicity, in this derivation the documents are all assumed to have the same length N. The derivation is equally valid if the document lengths vary.

According to the model, the total probability of the model is:

$$P(\boldsymbol{W},\boldsymbol{Z},\boldsymbol{\theta},\boldsymbol{\varphi};\alpha,\beta) = \prod_{i=1}^{K} P(\varphi_i;\beta) \prod_{j=1}^{M} P(\theta_j;\alpha) \prod_{t=1}^{N} P(Z_{j,t}|\theta_j) P(W_{j,t}|\varphi_{Z_{j,t}}),$$

where the bold-font variables denote the vector version of the variables. First, $\boldsymbol{\varphi}$ and $\boldsymbol{\theta}$ need to be integrated out.

$$\begin{aligned} P(\boldsymbol{Z},\boldsymbol{W};\alpha,\beta) &= \int_{\boldsymbol{\theta}} \int_{\boldsymbol{\varphi}} P(\boldsymbol{W},\boldsymbol{Z},\boldsymbol{\theta},\boldsymbol{\varphi};\alpha,\beta)\, d\boldsymbol{\varphi}\, d\boldsymbol{\theta} \\ &= \int_{\boldsymbol{\varphi}} \prod_{i=1}^{K} P(\varphi_i;\beta) \prod_{j=1}^{M} \prod_{t=1}^{N} P(W_{j,t} \mid \varphi_{Z_{j,t}})\, d\boldsymbol{\varphi} \int_{\boldsymbol{\theta}} \prod_{j=1}^{M} P(\theta_j;\alpha) \prod_{t=1}^{N} P(Z_{j,t} \mid \theta_j)\, d\boldsymbol{\theta}. \end{aligned}$$

https://en.wikipedia.org/wiki/Latent_Dirichlet_allocation

Monte Carlo Methods for Topic Models

- How do we actually *use* Monte Carlo for topic models?
- First we write out the posterior:

$$p(Z, \pi, \theta \mid X, \alpha, \beta) = \left[\prod_{i=1}^{n} p(\theta^i \mid \alpha) \prod_{j=1}^{d} p(z_j^i \mid \theta^i)\, p(x_j^i \mid z_j^i, \pi_j)\right]\left[\prod_{c=1}^{k} p(\pi_c \mid \beta)\right]$$

topics
word prob.
topic prop.
data (words)
prior on topic proportions
prior on word probabilities
topic proportion probability (document 'i')
topic probability (topic at position 'j' in document 'i')
word probability (word at position 'j' in document 'i')
word probability parameters (topic 'c')

Monte Carlo Methods for Topic Models

- How do we actually *use* Monte Carlo for topic models?
- First we generate samples from the posterior:
  - With Gibbs sampling we alternate between:
    - Sampling topics given word probabilities and topic proportions.
    - Sampling topic proportions given topics and prior parameters α.
    - Sampling word probabilities given topics, words, and prior parameters β.
  - Have a burn-in period, use thinning, try to monitor convergence, and so on.
- Then we use posterior samples to do inference:
  - Distribution of topic proportions for sample i is frequency in samples.
  - To see if words come from same topic, check frequency in samples.

Summary

- Hidden Markov models model time-series with hidden per-time cluster.
  - Inference with forward-backward; learn with EM.
  - Tons of applications; typically more realistic than Markov models.
  - Can make
- Topic models: latent-factor model of discrete data text.
  - The latent "factors" are called "topics".
- Latent Dirichlet allocation: hierarchical Bayesian topic model.
  - Represent words in documents as coming from different topics.
  - Each document has its own proportion for each topic.

- Next time: faster (but worse?) inference, variationally.

Outline

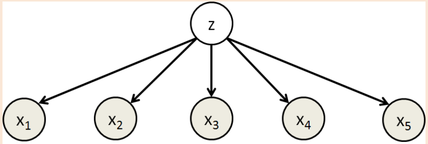
Mixture of Bernoullis Models

bonus!

- Recall the mixture of Bernoullis models:

$$p(x) = \sum_{c=1}^{k} p(z=c) \prod_{j=1}^{d} p(x_j \mid z=c).$$

- Given z, each variable x_j comes from a product of Bernoullis

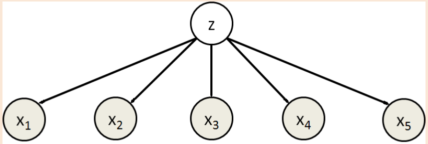


- This is enough to model *any* multivariate binary distribution.
  - But not an efficient representation: number of cluster might need to be huge.
    - Need to learn each cluster independently (no "shared" information across clusters).

Mixture of Independents as a UGM

bonus!

- The mixture of independents assumptions can be represented as a UGM:



  - "The x_j are independent given the cluster z".
- A log-linear parameterization for $x_j \in \{-1, +1\}$ and $z \in \{-1, +1\}$ could be

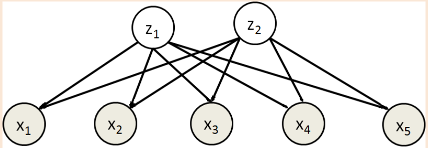
$$\phi_j(x_j) = \exp(w_j x_j), \quad \phi_z(z) = \exp(vz), \quad \phi_{j,z}(x_j, z) = \exp(w_j x_j z).$$

- We have three types of parameters:
  - Weight w_j in ϕ_j affects probability of $x_j = 1$ (independent of cluster).
  - Weight v in ϕ_z affecst probability that $z_j = 1$ (prior for cluster).
  - Weight w_j in $\phi_{j,z}$ affects probability that x_j and z are same.
    - Can encourage each binary variable to be same or different than "cluster sign".

"Double Clustering" Model

bonus!

- Now consider adding a second binary cluster variable:



  - "The x_j are independent given both cluster variables z_1 and z_2".
- A log-linear parameterization for $x_j \in \{-1, +1\}$ and $z_c \in \{-1, +1\}$ could be

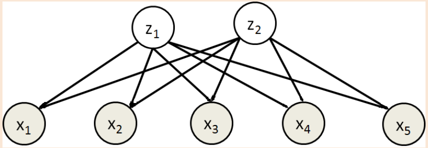
$$\phi_j(x_j) = \exp(w_j x_j), \quad \phi_c(z_c) = \exp(v_c z_c), \quad \phi_{j,c}(x_j, z_c) = \exp(w_{jc} x_j z)$$

- We have three types of parameters:
  - Weight w_j in ϕ_j affects probability of $x_j = 1$ (independent of cluster).
  - Weight v_c in ϕ_z affecst probability that $z_c = 1$ (prior for cluster variable).
  - Weight w_{jc} in $\phi_{j,z}$ affects probability that x_j and z_c are same.
    - Can encourage each binary variable to be same or different than "cluster variable".

"Double Clustering" Model

bonus!

- Now consider adding a second binary cluster variable:



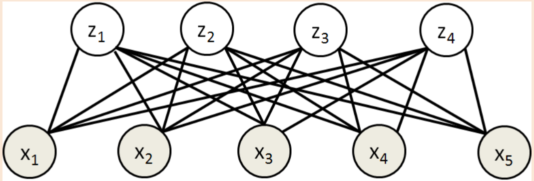
- Have we gained anything?
  - We have 4 clusters based on two hidden variables.
  - Each cluster shares parameters with 2 of the other clusters.

- Hope is to achieve some degree of composition
  - Don't need to re-learn basic things about the x_j in each cluster.
  - Maybe one hidden z_c models clusters, and another models correlations.
    - So that when you use both, you can capture both aspects.

Restricted Boltzmann Machines (RBMs)

bonus!

- Now consider adding two more binary latent variables:

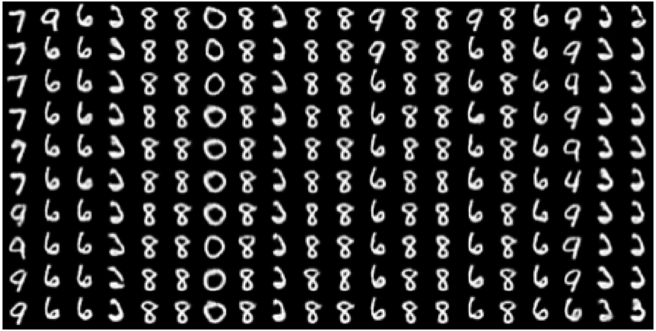


- Now we have 16 clusters, in general we'll have 2^k with k hidden binary nodes.
  - This discrete latent-factors give combinatorial number of mixtures.
    - You can think of each z_c as a "part" that can be included or not ("binary PCA").
- This is called a restricted Boltzmann machine (RBM).
  - A Boltzmann machine is a UGM with binary hidden variables.
- It is restricted because all edges are between "visible" x_j and "hidden" z_c.
  - If we know the x_j, then the z_c are independent.
  - If we know the z_c, then the x_j are independent.
  - Inference on both x and z is hard.
    - But we could alternate between Gibbs sampling of all x and all z variables.

Generating Digits with RBMs

bonus!

Here are the samples generated by the RBM after training. Each row represents a mini-batch of negative particles (samples from independent Gibbs chains). 1000 steps of Gibbs sampling were taken between each of those rows.

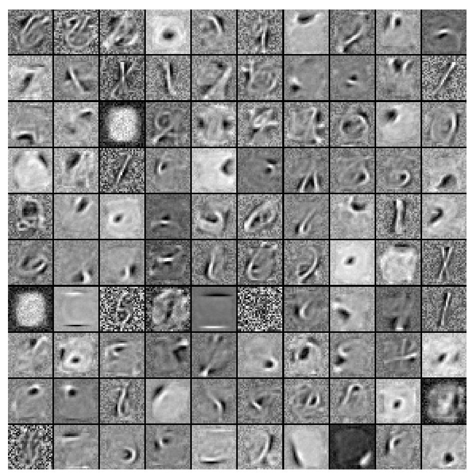


http://deeplearning.net/tutorial/rbm.html

Generating Digits with RBMs

bonus!

Visualizing each z_c's interaction parameters (w_{jc} for all j) as images:



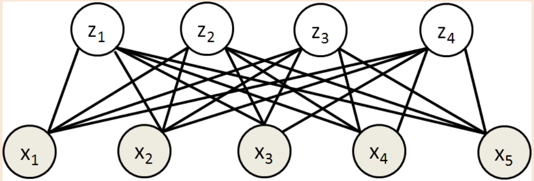
http://deeplearning.net/tutorial/rbm.html

Restricted Boltzmann Machines

bonus!

- The RBM graph structure leads to a joint distribution of the form

$$p(x,z) = \frac{1}{Z}\left(\prod_{j=1}^{d}\phi_j(x_j)\right)\left(\prod_{c=1}^{k}\phi_c(z_c)\right)\left(\prod_{j=1}^{d}\prod_{c=1}^{k}\phi_{jc}(x_j,z_c)\right).$$



- RBMs usually use a log-linear parameterization like

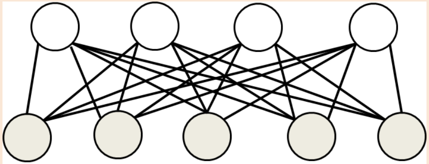
$$p(x,z) \propto \exp\left(\sum_{j=1}^{d} w_j x_j + \sum_{c=1}^{k} v_c z_c + \sum_{j=1}^{d}\sum_{c=1}^{k} w_{jc} x_j z_c\right),$$

for parameters w_j, v_c, and w_{jc} (variants exist for non-binary x_j).

Learning UGMs with Hidden Variables

bonus!

- For RBMs we have hidden variables:



- With hidden ("nuisance") variables z the observed likelihood has the form

$$p(x) = \sum_z p(x,z) = \sum_z \frac{\tilde{p}(x,z)}{Z}$$
$$= \frac{1}{Z} \underbrace{\sum_z \tilde{p}(x,z)}_{Z(x)} = \frac{Z(x)}{Z},$$

where $Z(x)$ is the partition function of the conditional UGM given x.

  - $Z(x)$ is cheap in RBMs because the z are independent given x.

Learning UGMs with Hidden Variables

bonus!

- This gives an observed NLL of the form

$$-\log p(x) = -\log(Z(x)) + \log Z,$$

where $Z(x)$ sums over hidden z values, and Z sums over z and x.

- The second term is convex but the first term is non-convex.
  - This is expected when we have hidden variables.

- With a log-linear parameterization, the gradient has the form

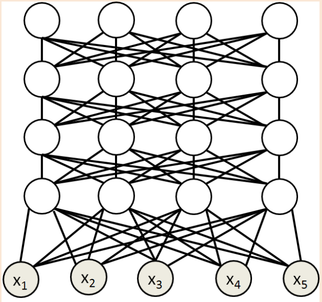
$$-\nabla \log p(x) = -\mathbb{E}_{z|x}[F(X,Z)] + \mathbb{E}_{z,x}[F(X,Z)].$$

- For RBMs, first term is cheap due to independence of z given x.
- We can approximate second term using block Gibbs sampling.
  - For other problems, you would also need to approximate first term.

Deep Boltzmann Machines

bonus!

- 15 years ago, a hot topic was "stacking RBMs", as in deep Boltzmann Machine:



- Part of the motivation for people to re-consider "deep" models.
- Model above allows block Gibbs sampling "by layer".
  - Variables in layer are conditionally independent given layer above and below.

Deep Boltzmann Machines

bonus!

- Performance of deep Boltzmann machine on NORB data:

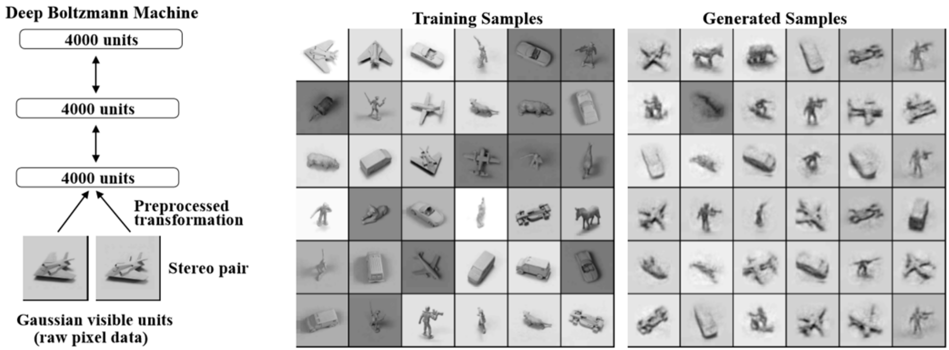


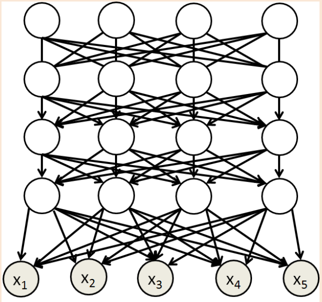
Figure 5: **Left:** The architecture of deep Boltzmann machine used for NORB. **Right:** Random samples from the training set, and samples generated from the deep Boltzmann machines by running the Gibbs sampler for 10,000 steps.

http://www.cs.toronto.edu/~fritz/absps/dbm.pdf

Deep Belief Networks

bonus!

- There were also deep belief networks where RBM outputs DAG layers.

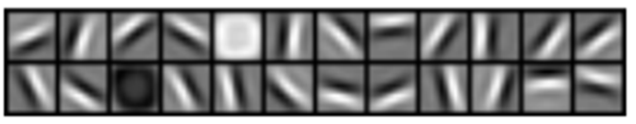


- More difficult to train and do inference due to explaining away.
- Though easier to sample using ancestral sampling.

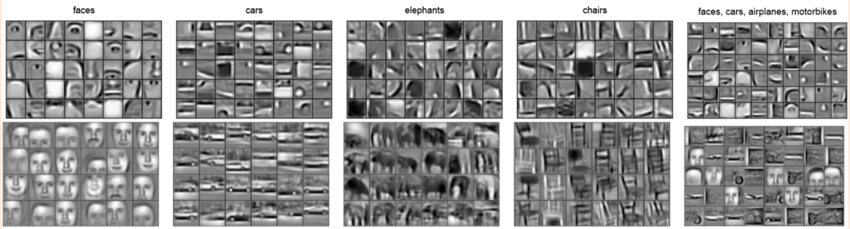
Cool Pictures Motivation for Deep Learning

bonus!

- First layer of z_i in a convolutional deep belief network:



- Visualization of second and third layers trained on specific objects:



http://www.cs.toronto.edu/~rgrosse/icml09-cdbn.pdf

- Many classes use these particular images to motivate deep neural networks.
  - But they're not from a neural network: they're from a deep DAG model.