COMS 4721: Machine Learning for Data Science
Lecture 18, 4/4/2017

Prof. John Paisley

Department of Electrical Engineering
& Data Science Institute

Columbia University

Another matrix factorization technique:

Measured elements in the matrix are all non-negative and also where we have to factorize that into a product of matrices that also have no negative entries in it.

TOPIC MODELING

MODELS FOR TEXT DATA

Motivation for models like LDA · is to model text.

Given text data we want to:

- Organize
- Visualize
- Summarize
- Search
- Predict
- Understand

Topic models allow us to

1. Discover themes in text
2. Annotate documents
3. Organize, summarize, etc.

TOPIC MODELING

Idea behind it: modelling different documents contain words in them that belong to different concepts.

The New York Times | http://nyti.ms/WO91dx

BUSINESS DAY

A Digital Shift on Health Data Swells Profits in an Industry

By JULIE CRESWELL FEB. 19, 2013

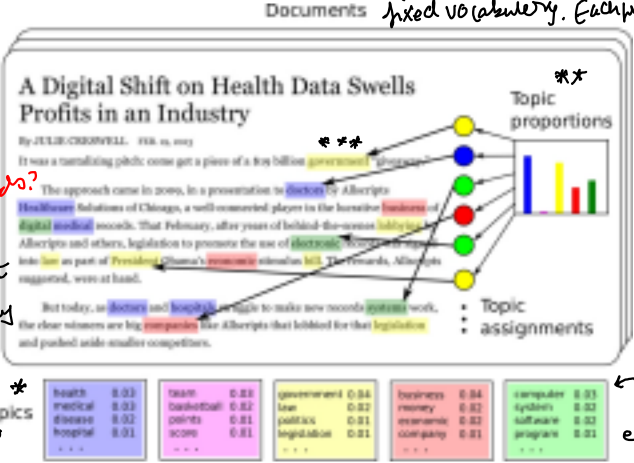
It was a tantalizing pitch: come get a piece of a $19 billion government "giveaway."

The approach came in 2009, in a presentation to doctors by Allscripts Healthcare Solutions of Chicago, a well-connected player in the lucrative business of digital medical records. That February, after years of behind-the-scenes lobbying by Allscripts and others, legislation to promote the use of electronic records was signed into law as part of President Obama's economic stimulus bill. The rewards, Allscripts suggested, were at hand.

But today, as doctors and hospitals struggle to make new records systems work, the clear winners are big companies like Allscripts that lobbied for that legislation and pushed aside smaller competitors.

TOPIC MODELING

Goal: to model all of this information. And what we assume starting is we have a set of probability distributions on a fixed vocabulary. Each prob. dist. has most of it on a subset of the words that somehow belong together and coherently express a theme.



Purpose of coloring words?

global thing that interacts with every document.

If you look at any list, every word appears on it.

A probabilistic topic model

- Learns distributions on words called “topics” shared by documents
- Learns a distribution on topics for each document
- Assigns every word in a document to a topic

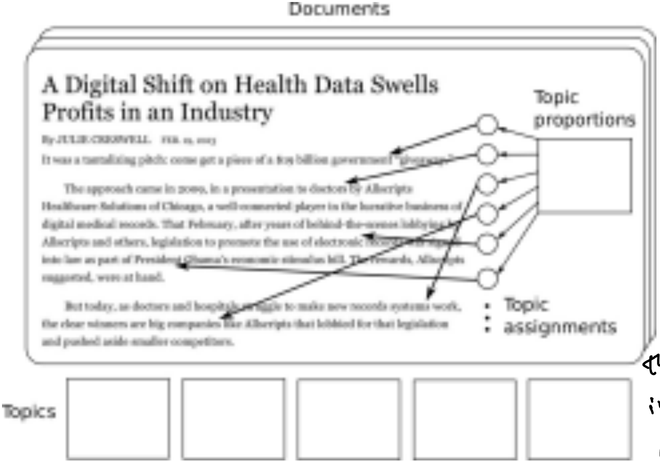
*If we view the prob. distribution as an ordered list. Of words based on how probable they are, we see that most probable words that come to the top all relate to the same thing.

Each topic captures one theme underlying the dataset.

** A vector of topic proportions. For ex: 5 topics → 5 dimensional prob. distribution on these topics. (So that's the 3rd document level variable.) why add up to 1?

*** For each word that appears in the document, an assignment of which topics it belongs to. Color coding for every single word that appears in the document.
Each word has to pick a topic that's going to come from. in this document.
And color coding acc. to that topic

TOPIC MODELING



that's the inference goal of a model like LDA.

However, none of these things are known in advance and must be learned

- Each document is treated as a “bag of words”
- Need to define (1) a model, and (2) an algorithm to learn it
- We will review the standard topic model, but won't cover inference

Goals:

1. Present generative process for LDA
2. Non-negative matrix factorization (more general technique for factorizing matrices

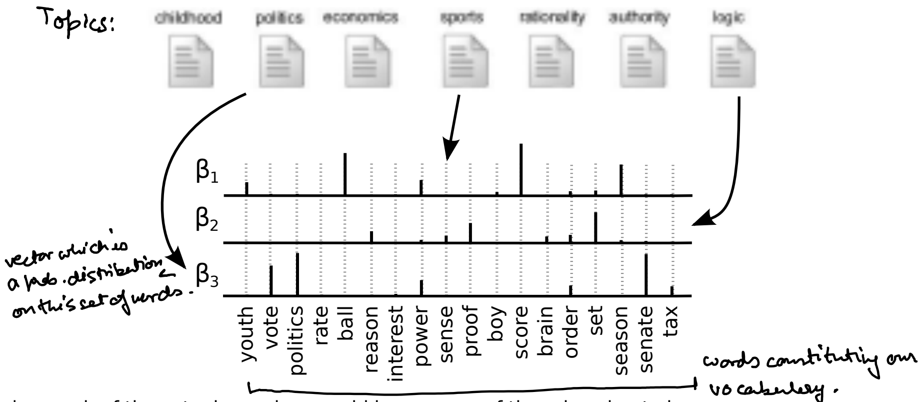
And

3. matrices and discuss to algorithms fordoing that that are relevant to the same problems that LDA addresses but are not equivalent to the algorithm that is used for LDA generally.

LATENT DIRICHLET ALLOCATION

There are two essential ingredients to latent Dirichlet allocation (LDA).

1. A collection of distributions on words (topics). why topic?
2. A distribution on topics for each document.



And so each of these topics and we could have many of them is going to be characterized by probability distribution on the same exact set of vocabulary words, but they're going to have different probability distributions.

There are two essential ingredients to latent Dirichlet allocation (LDA).

1. A collection of distributions on words (topics).
2. A distribution on topics for each document.

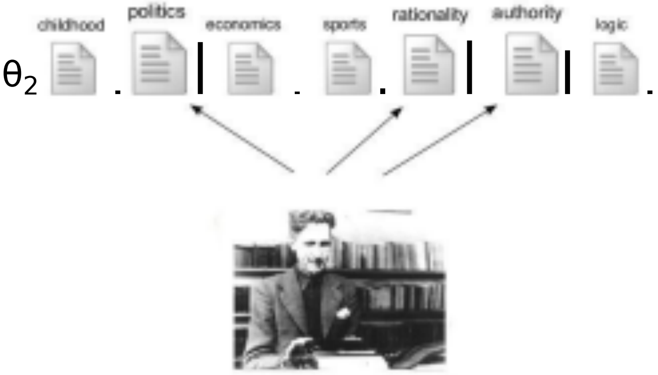
prob. distribution
picked by
the person

For a particular topic, we have a distribution on these topics

Latent Dirichlet allocation

There are two essential ingredients to latent Dirichlet allocation (LDA).

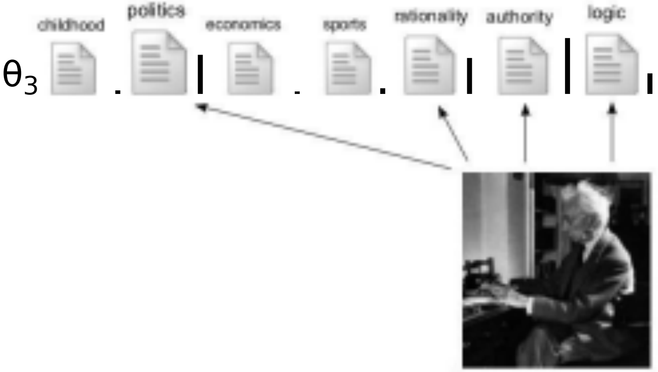
1. A collection of distributions on words (topics).
2. A distribution on topics for each document.



Latent Dirichlet allocation

There are two essential ingredients to latent Dirichlet allocation (LDA).

1. A collection of distributions on words (topics).
2. A distribution on topics for each document.



LATENT DIRICHLET ALLOCATION

There are two essential ingredients to latent Dirichlet allocation (LDA).

1. A collection of distributions on words (topics).
2. A distribution on topics for each document.

The generative process for LDA is:

done once.

1. Generate each topic, which is a distribution on words

Pick prior distribution of vectors that are non-negative and sum to 1. Default → Dirichlet

** d-dimensional (D words)

$$\beta_k \sim \text{Dirichlet}(\gamma), \quad k = 1, \ldots, K$$

[K different topics]
↳ K different prob. distributions on the same set of words.

2. For each document, generate a distribution on topics

how that document is going to split out its words across those topics.

*** K-dimensional (K topics)

$$\theta_d \sim \text{Dirichlet}(\alpha), \quad d = 1, \ldots, D$$

Generative process.

3. For the nth generated word in the dth document,

Decide which topic word comes from:

a) Allocate the word to a topic, $c_{dn} \sim \text{Discrete}(\theta_d)$ indicator

from a discrete distribution using the distribution on topics for the d^{th} document.

b) Generate the word from the selected topic, $x_{dn} \sim \text{Discrete}(\beta_{c_{dn}})$

document which word. uses the topic picked up by c_{dn}: 1 to K

* Representation Bayesian model like LDA:

1. We hypothesize a generative process for the data that we see
2. derive an inference algo. for doing the inverse problem of learning the actual parameters or a posterior distribution on these parameters that could explain the data the we saw.

** The prior model assumes that each of these topics is generated independently and identically distributed as a Dirichlet distribution.
$B_k \rightarrow$ the prior distribution placed on each of the k topics.
Each topic is generated once by drawing from this distribution and then fixed for all the time.

*** Then for each document we need to decide for that document how it's going to use the topics that are available to it.
Generated iid from the Dirichlet distribution. We do this once for each of D documents.

o Now that we have for D documents a distribution on the different themes that β captures, we have to generate the words that appear in that document.

oo So c_dn will pick out one of the capital K topics available to it, where the probability of picking a particular topic is encoded in this distribution vector Θ_d

(1) We don't know what any of these are except for the data x.
We don't know what c is for each word in each document.
So there are many of these indicators we have to learn.
We don't know what the distributions on topics are and
we also don't know what the topics themselves are.
So there are many different things we need to learn with this model.

(2) This is called a bag of words model.

So we're not modeling with LDA.
We're not modeling language as it appears in order.
We're simply taking all of the words that appear in the document and essentially throwing them in a bag and then jumbling them up, and modeling the entire set of words without any reference to order

No order taken into consideration in this generative process.

DIRICHLET DISTRIBUTION

→ prob. distribution on vectors that are non-negative and sum to 1.

A continuous distribution on discrete probability vectors. Let β_k be a probability vector and γ a positive parameter vector,

what is k?

$$p(\beta_k|\gamma) = \frac{\Gamma(\sum_v \gamma_v)}{\prod_{v=1}^{V} \Gamma(\gamma_v)} \prod_{v=1}^{V} \beta_{k,v}^{\gamma_v - 1}$$

β_k is a 10-dim vector

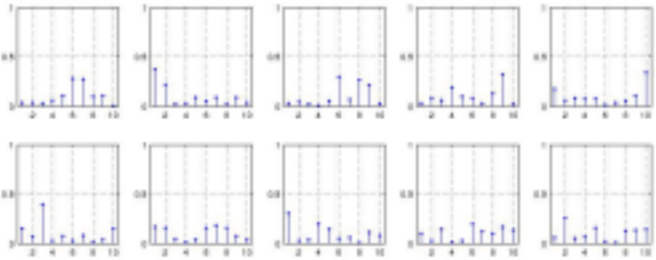
This defines the Dirichlet distribution. Some examples of β_k generated from this distribution for a constant value of γ and $V = 10$ are given below.

**

Generate 10 different random vectors from this distribution

$\gamma = 1$

10 dim vector, same value in each dimension.



[These are 10 different random variables all generated from a Dirichlet distribution with $\alpha=1$].

↓ they're all non-negative and sum to 1.

* $\beta_k \rightarrow$ if we let β_k be a random variable generated from a Dirichlet distribution what that means is: β_k is a vector of non-negative no.s that sum to 1.

So we can view this as a discrete probability distribution.

So in that sense the Dirichlet distribution is a prob. distribution on discrete distributions. ?

Ex: When we need to put a prior on a K or V-dimensional probability distribution, the Dirichlet is a natural choice.

? However, the Dirichlet itself is a continuous probability distribution. Dirichlet distribution is a continuous distribution on what can be viewed as a discrete probability distribution.

** Assumptions different values of this parameter are enforcing on this distribution:

For ex: if we assume the parameter to the Dirichlet distribution is shared, even though we have an index, V-dimensional prob. distribution in β. And so we need a V-dimensional parameter vector in γ. If we assume that all the values in the parameter vector are the same, we can then look at what different draws from this distribution will look like.

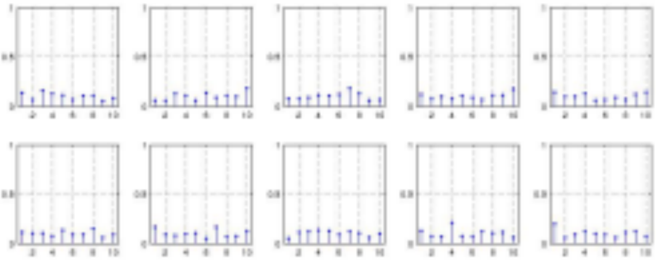
Dirichlet distribution

A continuous distribution on discrete probability vectors. Let β_k be a probability vector and γ a positive parameter vector,

$$p(\beta_k|\gamma) = \frac{\Gamma(\sum_v \gamma_v)}{\prod_{v=1}^{V}\Gamma(\gamma_v)} \prod_{v=1}^{V} \beta_{k,v}^{\gamma_v - 1}$$

This defines the Dirichlet distribution. Some examples of β_k generated from this distribution for a constant value of γ and $V = 10$ are given below.

$\gamma = 10$



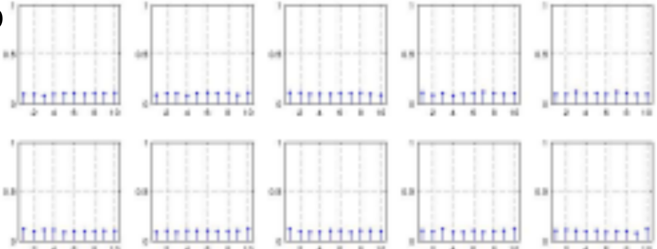
DIRICHLET DISTRIBUTION

A continuous distribution on discrete probability vectors. Let β_k be a probability vector and γ a positive parameter vector,

$$p(\beta_k|\gamma) = \frac{\Gamma(\sum_v \gamma_v)}{\prod_{v=1}^V \Gamma(\gamma_v)} \prod_{v=1}^{V} \beta_{k,v}^{\gamma_v - 1}$$

This defines the Dirichlet distribution. Some examples of β_k generated from this distribution for a constant value of γ and $V = 10$ are given below.

$\gamma = 100$



As $\alpha \uparrow$, from 10 to 100, the random variables that I generate start to look more and more uniform. So that's what the parameter does, it forces a more and more uniform distribution as this parameter goes to ∞.

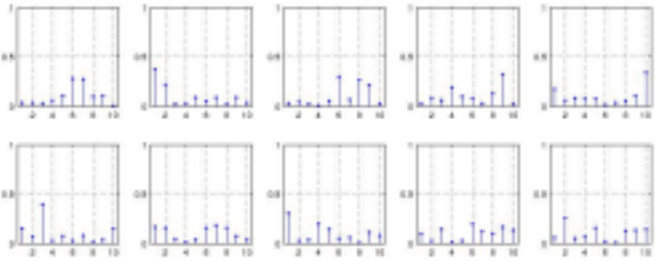
DIRICHLET DISTRIBUTION

A continuous distribution on discrete probability vectors. Let β_k be a probability vector and γ a positive parameter vector,

$$p(\beta_k|\gamma) = \frac{\Gamma(\sum_v \gamma_v)}{\prod_{v=1}^{V} \Gamma(\gamma_v)} \prod_{v=1}^{V} \beta_{k,v}^{\gamma_v - 1}$$

This defines the Dirichlet distribution. Some examples of β_k generated from this distribution for a constant value of γ and $V = 10$ are given below.

$\gamma = 1$



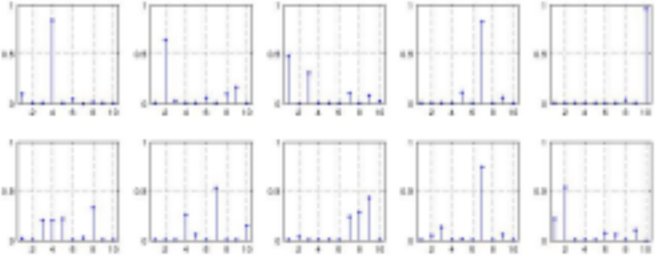
DIRICHLET DISTRIBUTION

A continuous distribution on discrete probability vectors. Let β_k be a probability vector and γ a positive parameter vector,

$$p(\beta_k|\gamma) = \frac{\Gamma(\sum_v \gamma_v)}{\prod_{v=1}^{V}\Gamma(\gamma_v)} \prod_{v=1}^{V} \beta_{k,v}^{\gamma_v - 1}$$

This defines the Dirichlet distribution. Some examples of β_k generated from this distribution for a constant value of γ and $V = 10$ are given below.

$\gamma = 0.1$



Why it becomes less sparse?

3) I drop from 1 to 0, I get more and more distribution which put most of its probability mass on a smaller and smaller subset of dimensions.

DIRICHLET DISTRIBUTION

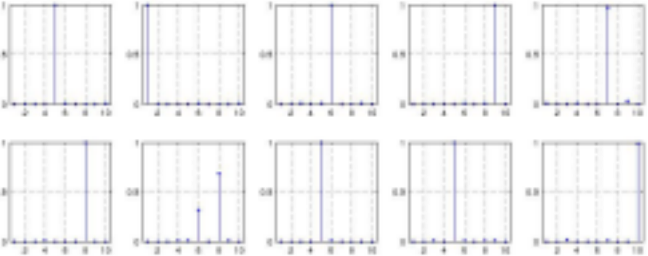
A continuous distribution on discrete probability vectors. Let β_k be a probability vector and γ a positive parameter vector,

In practice, with LDA, these parameter will set to <1, which enforces our priori belief that each topic should only put its prob. mass on a small subset of the total no. of words available to it.

$$p(\beta_k|\gamma) = \frac{\Gamma(\sum_v \gamma_v)}{\prod_{v=1}^{V} \Gamma(\gamma_v)} \prod_{v=1}^{V} \beta_{k,v}^{\gamma_v - 1}$$

This defines the Dirichlet distribution. Some examples of β_k generated from this distribution for a constant value of γ and $V = 10$ are given below.

$\gamma = 0.01$



Similarly for the the prior distribution and on the topic proportions is discrete as well.

And so if we let alpha be something that's less than one,
that enforces our apriori belief that each document
only uses a subset of the different topics available to it.
Ex:
So if there are 30 different things a document could talk about represented
by 30 different topics represented by 30 different probability vectors beta.
We might set the prior alpha such that only five or
an expectation only five for example would be manifested in any individual document.

LDA OUTPUT (LDA to 2 million topics. We learned the topics, and showing ten of these topics)

each is one vector β_k over 8000 word vocabulary.

10 most probable vectors are to β_1

? This is not something that's encoded in the model at all. This is something that's picked out in the posterior distribution because that information is contained in the data.

LDA outputs two main things:

1. A set of distributions on words (topics). Shown above are ten topics from NYT data. We list the ten words with the highest probability.
2. A distribution on topics for each document (not shown). This indicates its thematic breakdown and provides a compact representation.

ls very closed related to

for a particular document, what is the prob. that the n^{th} word in the d^{th} document is equal to the index i. We have a vocabulary list in which each word is indexed by i, that's what we're querying here.

Q: For a particular document, what is $P(x_{dn} = i|\beta, \theta_d)$?

A: Find this by integrating out the cluster assignment,

Given only $\beta \to$ set of topics
$\theta_d \to$ only the distribution on these topics for d^{th} document *

$$\begin{aligned} P(x_{dn} = i|\boldsymbol{\beta}, \theta) &= \sum_{k=1}^{K} P(x_{dn} = i, c_{dn} = k|\boldsymbol{\beta}, \theta_d) \quad ** \\ &= \sum_{k=1}^{K} \underbrace{P(x_{dn} = i, |\boldsymbol{\beta}, c_{dn} = k)}_{= \beta_{ki}} \underbrace{P(c_{dn} = k|\theta_d)}_{= \theta_{dk}} \quad *** \end{aligned}$$

likelihood   prior

V x k matrix   K x d matrix

Let $B = [\beta_1, \ldots, \beta_K]$ and $\Theta = [\theta_1, \ldots, \theta_D]$, then $P(x_{dn} = i|\beta, \theta) = (B\Theta)_{id}$

V for V vocabulary words

taking each distribution on topics and putting them along a column.

In other words, we can read the probabilities from a matrix formed by taking the product of two matrices that have nonnegative entries.

* Notice, there is no indicator here of which of the topics this word came from, C is gone. ?

In other words, this is the marginal distribution where we have integrated out C. So we'll write that here & see if we can calculate that.

** a sum of over the joint distribution of the n^{th} word in the d^{th} document being equal to i and the assignment of n^{th} word in the d^{th} document to cluster k.

And now we have a joint distribution over this word index, and also the topic assignment for that word.

We sum out the topic assignment, and of course that integral or that sum is equal to marginal distribution we're interested in.

*** Break this joint distribution over these 2 things into:

1. a conditional distribution of x given c and β. And because we have c, we don't need to know Θ anymore. So x is conditionally independent of Θ given C. (That's why Θ doesn't appear here.)

2. Prior on the topic assignment given the vector Θ

o The probability of word i given that word comes from topic k is just equal to the i^{th} dimension of topic vector β_K.

oo Prior probability of any word in the d^{th} document coming from topic k is just equal to the k^{th} dimension of d^{th} probability vector Θ_D, So we have Θ_{dk}.

☆ We can also read off this probability by this matrix product of $B \times \Theta$ and getting all probabilities for all the words in all documents. And then reading off id^{th} entry (probability of seeing word i in document d.)

In a sense, topic modelling can be thought of as a non-negative matrix factorization. We
We want to learn the factorization of this matrix of non-negative probabilities, and represent that as equal to a product of 2 matrices that also have non-negative values..

NONNEGATIVE MATRIX FACTORIZATION

NONNEGATIVE MATRIX FACTORIZATION (Discuss 2 other methods for doing NMF. (and actual algos))

LDA can be thought of as an instance of nonnegative matrix factorization.

- It is a probabilistic model.
- Inference involves techniques not taught in this course.

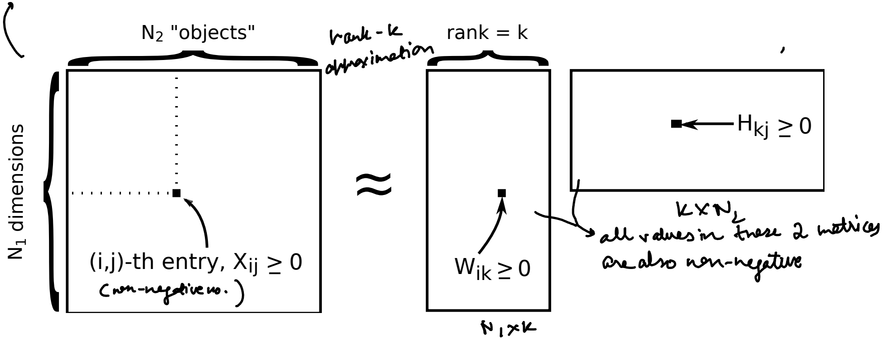
We will discuss two other related models and their algorithms. These two models are called *nonnegative matrix factorization* (NMF)

- They can be used for the same tasks as LDA
- Though "nonnegative matrix factorization" is a general technique, "NMF" usually just refers to the following two methods.

↳ taking a non-negative matrix and factorizing it into a product of 2 non-negative matrices.

NONNEGATIVE MATRIX FACTORIZATION

Assume we have all values in this matrix. (Last time we had many missing values in this matrix)



We use notation and think about the problem slightly differently from PMF

- Data X has nonnegative entries. None missing, but likely many zeros.
- The learned factorization W and H also have nonnegative entries.
- * The value $X_{ij} \approx \sum_k W_{ik}H_{kj}$, but we won't write this with vector notation
- Later we interpret the output in terms of columns of W and H.

*So notice that last time also, the way that we wrote this mathematically was to say that, each row here and each column was one vector.
And then we represented one particular entry, as a dot product of two vectors.
In this case we're not going to write that.
So it's purely a notational thing.

$$X_{ij} = \sum_k W_{ik} H_{kj}$$

NONNEGATIVE MATRIX FACTORIZATION

(using nmf)

What are some data modeling problems that can constitute X?

- Text data:
  - Word term frequencies
  - Data matrix X X_{ij} contains the number of times word i appears in document j.

  And then we factorize that matrix.
- Image data:
  - Face identification data sets
  - * Put each *vectorized* $N \times M$ image of a face on a *column* of X.
- Other discrete grouped data:
  - Quantize *continuous* sets of features using K-means
  - X_{ij} counts how many times group j uses cluster i.
  - ** For example: group = song, features = $d \times n$ spectral information matrix

each song would be 1 document. For each song, which is one-long audio signal we extract a set of features that are d-dimensional

*

In this case what we do is, we take an n by m image.
So all the images have to be exactly the same size.
We vectorize it, by taking the two-dimensional values,
and using the same exact method for each image.
We put them all into one long vector of size n times m.
And then we put each image along one column, of the data matrix x.

**

For each song, which is one long audio signal, we extract a set of features that are d-dimensional, each feature is dimensional. And imagine that we have n of these individual features, that are spaced out in time across the song.
So this could be, for example, something like the spectral information in the signal.
Where d would be the frequency content.
n would index at time slice.
And we have a bunch of these features extracted,
across the song at different time points, saying this is the spectral,
the frequency content, which will capture things like, guitars and voices and so on.
We then quantize all of those features, for
all the songs together at the same time using k means clustering.
Where the k now corresponds to the vocabulary,
the size of the vocabulary, so we set k.
Then every song, is now reduced from a set of d-dimensional features,
to a sequence of indexes of which cluster, each of those features was assigned to.
And then we histogram that, put that along a column of the matrix x, and factorize it.
And that way we can say, how song was used the code book,
how they cluster across the code book.
And so similar songs, should use the codes from k means in a similar way, and so on.

specific instance of non-negative matrix factorization

NMF minimizes one of the following two objective functions over W and H.

Choice 1: Squared error objective

approximate X as a product of non-negative matrix W and H.

take each element square it and sum it up entirely

$$\|X - WH\|_f^2 = \sum_i \sum_j (X_{ij} - (WH)_{ij})^2$$

frobenius norm.

one objective we will try to minimize using NMF algo. So, this will be one of 2 NMF algos.

Choice 2: Divergence objective

also a distance measure b/w X & WH

$$D(X\|WH) = -\sum_i \sum_j [X_{ij} \ln(WH)_{ij} - (WH)_{ij}]$$

different algorithm

sum over all elements in data matrix

- Both have the constraint that W and H contain nonnegative values.
- NMF uses a fast, simple algorithm for optimizing these two objectives.

MINIMIZATION AND MULTIPLICATIVE ALGORITHMS[1]

Recall what we should look for in minimizing an objective "$\min_h F(h)$":

1. A way to generate a sequence of values $h^1, h^2, \ldots,$ such that

$$F(h^1) \geq F(h^2) \geq F(h^3) \geq \cdots$$

evaluation of objective is monotonically decreasing in this sequence.

NMF does this by introducing some auxiliary function.

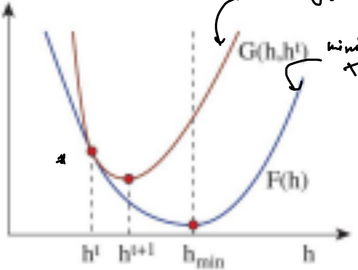
2. Convergence of the sequence to a local minimum of F

The following algorithms fulfill these requirements. In this case:

- Minimization is done via an "auxiliary function."
- Leads to a "multiplicative algorithm" for W and H.
- We'll skip details (see reference).

auxiliary function

minimize this



[1] For details, see D.D. Lee and H.S. Seung (2001). "Algorithms for non-negative matrix factorization." *Advances in Neural Information Processing Systems.*

* we introduce at each iteration a function that approximates it, is equal to the function at iteration h_t, is convex and also bounded below by the original function.

Instead of minimizing in iteration t this blue function, we minimize this red function. And so we can guarantee that we get a better value. → how?

Similarity:

red function. T is like the f^n L.

gap → KL divergence

blue function → marginal likelihood

MULTIPLICATIVE UPDATE FOR $\|X - WH\|^2$

Problem (minimize sum of squared errors)

$$\min \sum_{ij}(X_{ij} - (WH)_{ij})^2 \qquad \text{subject to } W_{ik} \geq 0,\ H_{kj} \geq 0.$$

Algorithm

- Randomly initialize H and W with nonnegative values.
- Iterate the following, first for all values in H, then all in W: (order does not matter.)

Proof: By iteratively updating H & W in this way, we're continually finding new values for H and W that are materially decreasing this objective f^{n}.

old value

gain

$$H_{kj} \leftarrow H_{kj} \frac{(W^T X)_{kj}}{(W^T WH)_{kj}},$$

$$W_{ik} \leftarrow W_{ik} \frac{(XH^T)_{ik}}{(WHH^T)_{ik}},$$

Why called multiplicate update? Every update to each value in H and W is found by taking the old value and multiplying it by something.

this something is a fn of data matrix X, most recent value of W and H.

And we just pick out the correct elements in these products.

until the change in $\|X - WH\|^2$ is "small."

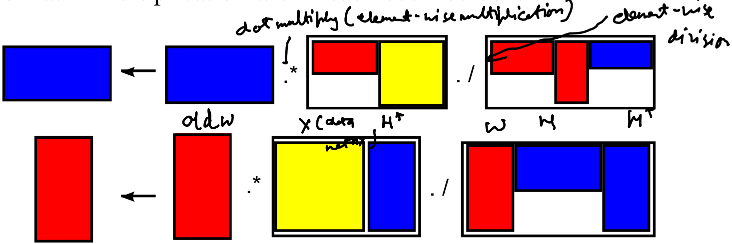
Similarly for the update of w, we take some matrix products, pick out the correct elements, multiply it by the original value to get the new value.

VISUALIZATION AND MAXIMUM LIKELIHOOD

In practice we can actually the entire all at the same time in this way.

A visualization that may be helpful. Use the color-coded definition above.

- Use element-wise multiplication/division across three columns below.
- Use matrix multiplication within each outlined box.



Probabilistically, the squared error penalty implies a Gaussian distribution,

MLE for a model where the elements of X are Gaussian with this mean and this variance. doesn't matter.

$$X_{ij} \sim N(\textstyle\sum_k W_{ik}H_{kj}, \sigma^2)$$ subject to parameter constraint that $W, H \geq 0$.

Since $X_{ij} \geq 0$ (and often isn't continuous), we are making an incorrect modeling assumption. Nevertheless, as with PMF it still works well.

So this is like what we're doing, Squared error penalty is trying to minimize the log of the Gaussian penalty.

MULTIPLICATIVE UPDATE FOR $D(X\|WH)$

Minimizing the divergence penalty. look at NMF updates that result from NMF derivation for this objective

Problem

objective

$$\min \sum_{ij} \left[X_{ij} \ln \frac{1}{(WH)_{ij}} + (WH)_{ij} \right] \quad \text{subject to} \quad W_{ik} \geq 0, \; H_{kj} \geq 0.$$

evaluate this objective f^n. It will be monotonically decreasing.

non-negativity constraints

Algorithm

- Randomly initialize H and W with nonnegative values.
- Iterate the following, first for all values in H, then all in W:

old value ← gain

$$H_{kj} \quad \leftarrow \quad H_{kj} \frac{\sum_i W_{ik} X_{ij} / (WH)_{ij}}{\sum_i W_{ik}},$$

old value ← gain

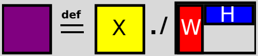
$$W_{ik} \quad \leftarrow \quad W_{ik} \frac{\sum_j H_{kj} X_{ij} / (WH)_{ij}}{\sum_j H_{kj}},$$

This W is the old value.

Iterate between these updates, and evaluate the objective f^n, it will be monotonically decreasing, and we terminate this algorithm when the improvement is relatively small.

until the change in $D(X\|WH)$ is "small."

VISUALIZATION



*(end)

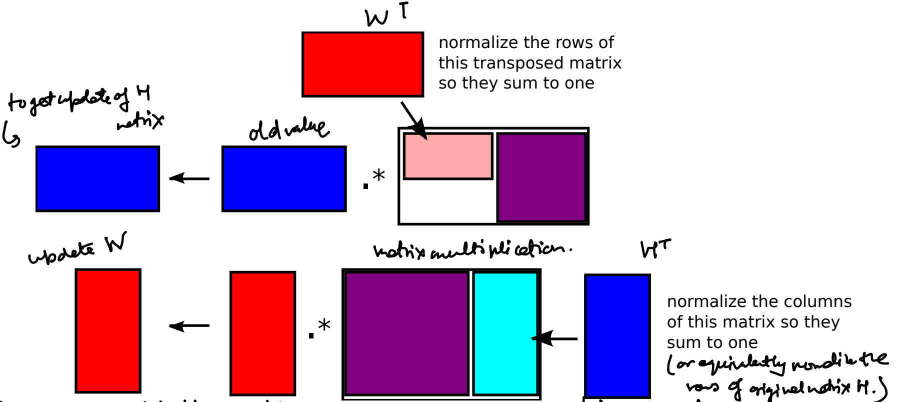
So, the algorithm would look something like iterating between updating this matrix purple, updating blue, going back to update this matrix, and then updating red.

data matrix element-wise divided by it's approximation.

Visualizing the update for the divergence penalty is more complicated.

[WH is the approximation of X.]

- Use the color-coded definition above.
- "Purple" is the data matrix "dot-divided" by the approximation of it.



=> When we update H, we need to update purple matrix because H has changed, and hence the elementwise division has changed. *

What is the equivalent probabilistic model that we have that would lead to the same exact update rules as we have with the divergence penalty, with this algorithm.

(Probabilistic)

The maximum likelihood interpretation of the divergence penalty is more interesting than for the squared error penalty.

discrete distribution on non-negative nos
x that is Poisson distributed, can takes values from {0, 1 ... ∞}.

If we model the data as independent Poisson random variables

Generative process:

$E(x) = \lambda$ [Matrix factorization we're doing is like learning the expected value for the data matrix under a Poisson generation assumption]

parameter

$$X_{ij} \sim \text{Pois}((WH)_{ij}), \qquad \text{Pois}(x|\lambda) = \frac{\lambda^x}{x!}e^{-\lambda}, \;\; x \in \{0, 1, 2, \dots\},$$

Poisson random variable with parameter ij^{th} element of our data matrix

then the negative divergence penalty is maximum likelihood for W and H.

$$
\begin{aligned}
-D(X\|WH)_{*} &= \sum_{ij} [X_{ij} \ln(WH)_{ij} - (WH)_{ij}] \\
&= \sum_{ij} \ln P(X_{ij}|W, H) + \text{constant}
\end{aligned}
$$

adding some constant, is equal to the negative of divergence penalty we're trying to minimize.

(to take care of terms: $x!$)

independence assumption

log

A product over Poisson random variables, where we're evaluating at pt X_{ij}, using parameter equal to the ij^{th} element of the matrix product WH.

We use: $P(X|W, H) = \prod_{ij} P(X_{ij}|W, H) = \prod_{ij} \text{Pois}(X_{ij}|(WH)_{ij})$.

likelihood of each element of X given W and H.

do MLE on this

* We want to minimize the divergence penalty, that equivalent to maximizing the negative of the divergence, which equals $-D(X \| WH)$ which is equivalent to maximising the likelihood of this Poisson model.

So, this makes a bit more sense, for example.
If we're trying to factorize matrices with counts in them,
term frequency matrices to do topic modeling.
The data matrix has energy value counts.
And a Poisson is a natural model for that, because the counts could be zero through
zero, one, two, up to, there's no limit to how many counts there could be.
However, its very low probability to have high counts,
which can be captured by this distribution.
And so, we're doing maximum likelihood for
a model that fits the type of data that we're seeing.
So, in a sense, this penalty makes more interpretive sense.

NMF AND TOPIC MODELING (How we can relate NMF to topic modelling)

As discussed, NMF can be used for topic modeling. In fact, one can show that the divergence penalty is closely related mathematically to LDA.

The way we can do topic modelling with NMF using this divergence penalty:

arbitrary indexing of our documents

Step 1. Form the term-frequency matrix X. ($X_{ij} = \#$ times word i in doc j)

arbitrary indexing of our vocabulary

Step 2. Run NMF to learn W and H using $D(X\|WH)$ penalty

Step 3. As an added step, after Step 2 is complete, for $k = 1, \ldots, K$

1. Set $a_k = \sum_i W_{ik}$ k^{th} column of matrix W
2. Divide W_{ik} by a_k for all i
3. Multiply H_{kj} by a_k for all j

If we do this all we have done is scale our data. We haven't changed the product of our data, WH.

Notice that this is does not change the matrix multiplication WH.

But now we can interpret the columns of W as probability distributions on words in a vocabulary, and so we can interpret the columns of W as topics.

Interpretation: The kth *column* of W can be interpreted as the kth *topic*. The jth *column* of H can be interpreted as how much document j uses each topic. ?

We can now interpret the columns of W as probability distributions on words in a vocabulary, so as topics. And therefore if we factorize a data matrix consisting of counts of no. of occurrences of words & documents. We can view the ordering of rows of each column of W as topic, learnt meaningful topics this way.

So, if we do this, all we've done is is scaled our data,
but we haven't changed the multiplication, the product, of our data, WH.

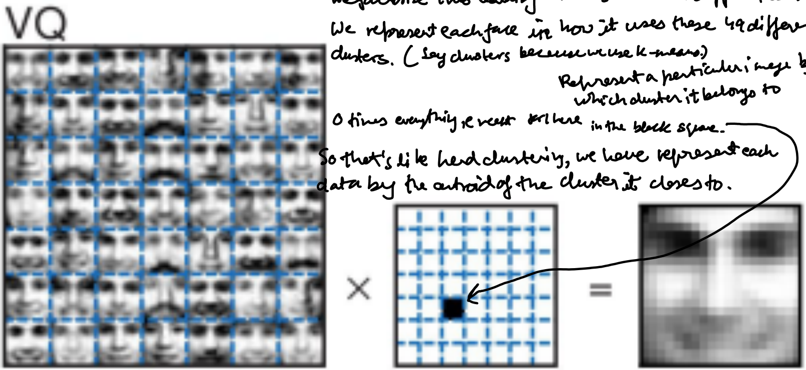
Also, if we think in terms of maximum likelihood, and in terms of divergence minimization
notice that both of those penalty terms don't penalize the actual values of W and H.
All they do is look at the resulting values in this matrix product.
So by doing this additional step we have not changed the maximum likelihood solution that we
found or the the minimum of this divergence bcoz they only depend on the product.

NMF and Face Modeling

each column of X consists of different images of face, and each row corresponds to a pixel in the original image.

For face modeling, put the face images along the columns of X and factorize. Show columns of W as image. Compare this with K-means and SVD.

We factorize this learning 7×7, about 49 different clusters.
We represent each face in how it uses these 49 different clusters. (Say clusters because we use k-means)
Represent a particular image by which cluster it belongs to
0 times everything except for here in the black square.
So that's like hard clustering, we have represent each data by the centroid of the cluster it closes to.



K-means (i.e., VQ): Equivalent to each column of H having a single 1.
K-means learns averages of full faces. View as matrix factorization, every data point can be cast to be matrix of factor loadings like this matrix.

? is it it 1-hot encoding? How does it consider avg. ?

For face modeling, put the face images along the columns of X and factorize. Show columns of W as image. Compare this with K-means and SVD.

mean of all faces

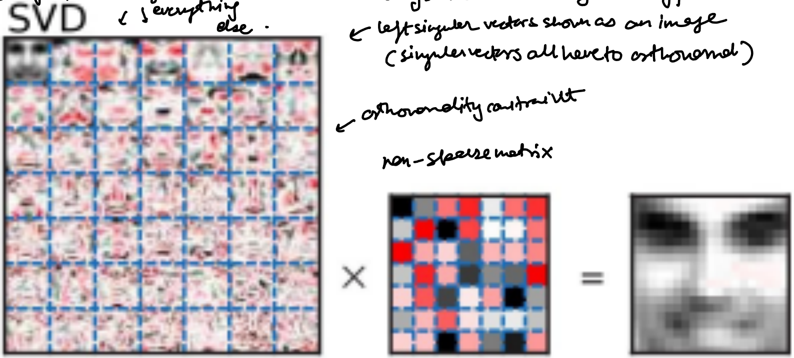
singular values of & everything else.

We took SVD of same data matrix of columns of faces.

← left singular vectors shown as an image (singular vectors all have to orthonormal)

← orthonormality constraint

non-sparse matrix



SVD: Finds the singular value decomposition of X.
Results not interpretable because of $\pm$ values and orthogonality constraint

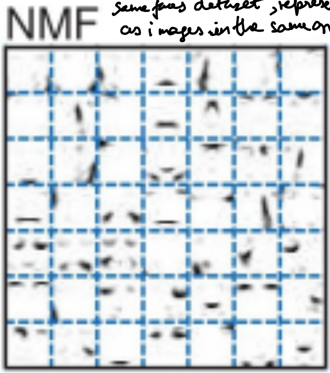
NMF and Face Modeling

Can learn something like, we don't learn it with SVD. because it has non-negativity constraints, which enforces this part-based learning.

For face modeling, put the face images along the columns of X and factorize. Show columns of W as image. Compare this with K-means and SVD.

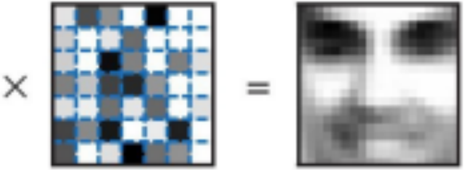
What different columns of W look like on the same faces dataset, represent those columns of W as images in the same original space we get something like this.

And so in this case NMF will often be described as being learning a parts based representation because for faces. We take all the faces and we learn different parts of those faces and represent each face as some linear combination of those parts.



each of these images picks out a part of the face.

weighted sum of different parts of these faces that learn from the dataset



NMF learns a "parts-based" representation. Each column captures something interpretable. This is a result of the nonnegativity constraint.

It's taking the data and finding the different aspects of the data that underlie it and then representing each data point as a sum of those aspects