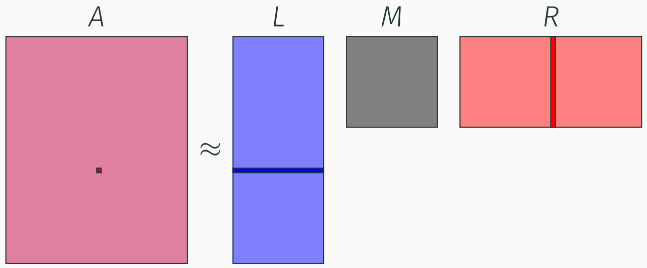
Numerical Methods for Data Science: Latent Factor Models, Part II

David Bindel

17 June 2019

Department of Computer Science
Cornell University

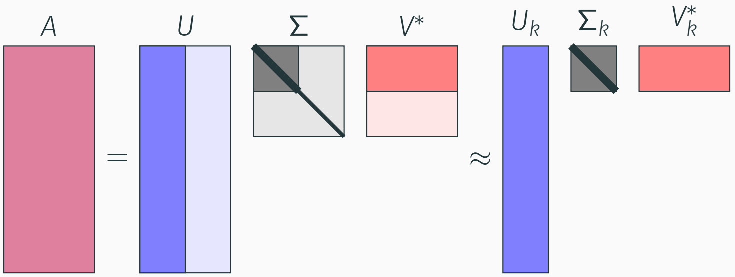
Reminder: Latent Factor Modeling



Underdetermined without constraints on L, M, and R.

- Simplest: Orthonormality constraints (SVD)
- Harder: L or R drawn from A (ID, CUR)
- Hardest: **Non-negativity and sparsity constraints**

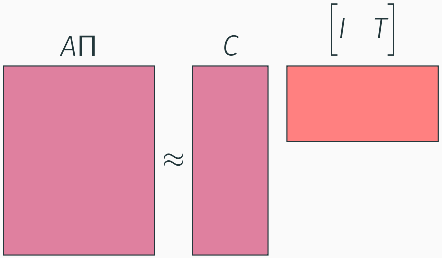
Reminder: Latent Factor Modeling and SVD



Simplest: Orthonormality constraints (SVD and the like)

- Straightforward to compute
    - Lots of good codes to use (sparse or dense case)
    - Existence proof is a greedy algorithm!
- Provides useful comparisons for indexing, clustering, etc
- Factors (latent coordinates) are not easy to interpret

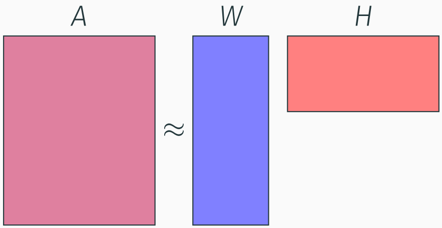
Reminder: Latent Factor Modeling and ID/CUR



Harder: Subset constraints (factors from rows/cols of A)

- Good selection of rows/cols gives
  - Nice factors (e.g. $|t_{ij}| \leq 2$)
  - Approximation error bounds
- Good building block: pivoted QR (greedy approach)
  - Greedy and suboptimal (though quality bounds exist)
  - Provides a good initial guess (refine by swapping)

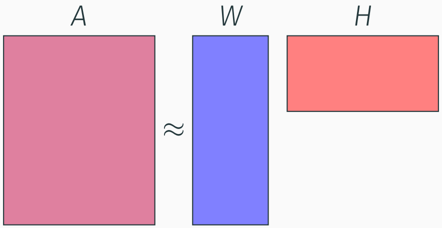
Latent Factor Modeling and Non-Negativity



Hardest: Non-negativity constraints ($W \in \mathbb{R}_{+}^{m\times k}, H \in \mathbb{R}_{+}^{k\times n}$)

- A hard optimization problem!
  - Standard algorithms converge to local minima
  - Few nontrivial approximation error bounds
- But this is great for interpretability!

Interpreting Non-Negative Matrix Factorizations (NMF)



Domain	Object ($a_{:,j}$)	Cluster ($w_{:,k}$)
Document	Word	Topic
Image	Pixel	Segment
Network	User	Community
Legislature	Member	Party/Group
Playlist	Song	Genre
Chemical spectra	Mixture	Molecules

Why is it Hard?

Non-negative matrix factorization is hard!

- The role of k is very unclear
- The problem is nonconvex with many local minima
- Greedy strategies are very suboptimal

But: sometimes it seems easier in practice!

Can deal with this in two ways

- Use locally-convergent iterations
  (and maybe think about initialization)
- Find structural reasons for non-hardness

We will tackle both.

The Optimization Problem

Goal:

$$A \approx WH$$

Several options to quantify "$\approx$"; we use Frobenius norm:

$$\text{minimize } \frac{1}{2}\|A - WH\|_F^2 \text{ where } W \in \mathbb{R}_+^{m\times k}, H \in \mathbb{R}_+^{k\times n}$$

Getting the Gradient

Note $\|\cdot\|_F$ is a Euclidean norm for the inner product

$$\langle X, Y\rangle_F = \mathrm{tr}(Y^T X)$$

Compute variations (directional derivatives) of

$$\phi(W, H) = \|R\|_F^2 = \|A - WH\|_F^2$$

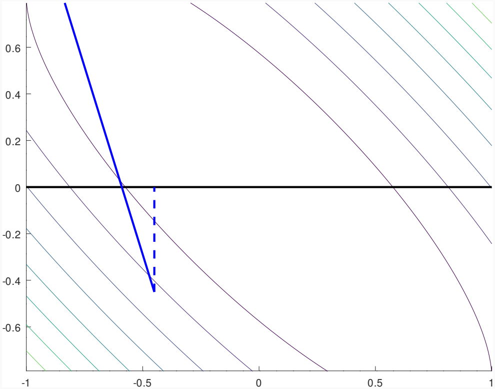
as follows:

$$\delta\phi = \delta\left[\frac{1}{2}\langle R, R\rangle_F\right] = \langle \delta R, R\rangle_F = -\langle (\delta W)H, R\rangle_F - \langle W(\delta H), R\rangle_F.$$

Do some algebra (cyclic invariance of traces) to rearrange as

$$\delta\phi = -\langle \delta W, RH^T\rangle_F - \langle \delta H, W^T R\rangle_F$$

Projected Gradient Descent



$$x^{k+1} = \mathcal{P}\left(x^{k+1} - \alpha_k \nabla \phi(x^k)\right)$$

Projected Gradient Descent for NMF

Given an objective ϕ and a projection $\mathcal{P}$ onto a convex constraint set, projected gradient descent iteration is

$$x^{k+1} = \mathcal{P}\left(x^{k+1} - \alpha_k \nabla\phi(x^k)\right)$$

In our case: $\mathcal{P}(X) = [X]_+$ is elementwise max with zero, and

$$\delta\phi = -\langle \delta W, RH^T \rangle_F - \langle \delta H, W^T R \rangle_F$$

so

$$\begin{aligned} W^{\mathrm{new}} &= [W + \alpha R H^T]_+ \\ H^{\mathrm{new}} &= [H + \alpha W^T R]_+ \end{aligned}$$

This converges for appropriate choices of α.

Multiplicative Updates

Can also choose more than one step size in PGD:

$$W^{\mathrm{new}} = [W + S \odot (RH^T)]_+$$
$$W^{\mathrm{new}} = [H + S' \odot (W^T R)]_+$$

Choose non-negative step sizes

$$S = W \oslash (WHH^T), \quad S' = H \oslash (W^T WH).$$

With these choices, terms cancel, so that

$$W^{\mathrm{new}} = S \odot (AH^T) = W \oslash (WHH^T) \odot (AH^T)$$
$$H^{\mathrm{new}} = S' \odot (W^T A) = H \oslash (W^T WH) \odot (W^T A).$$

Scaling is such that we no longer have to project.
Guaranteed non-increasing – but no convergence guarantee.

This is the Lee and Seung scheme.

Death by a Million Steps

Pete Stewart on “long run” convergence of gradient descent:

> *But this* long run *is a misleading guide to current affairs.* In the long run *we are all dead.*
>
> — *John Maynard Keynes (1923)*

Issues:

- Each step is expensive (compute with full matrices)
- We may need a lot of steps
- We need some smart way to choose α (for PGD)

Can we do better? Maybe with cheaper steps?

One Approach: Stochastic Gradient Descent

Projected gradient descent iteration:

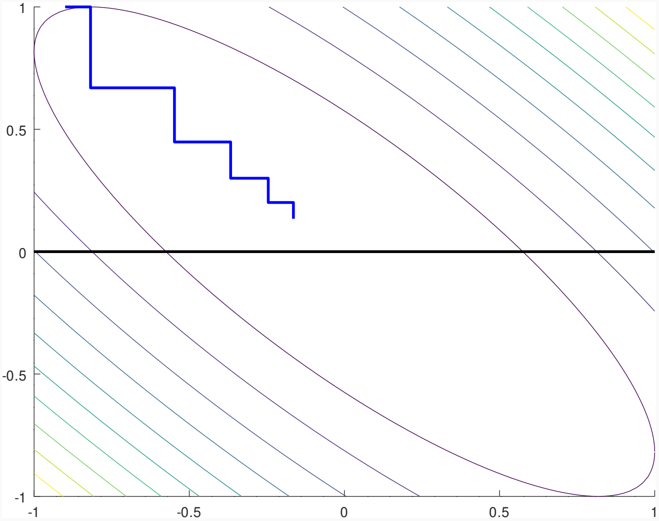
$$x^{k+1} = \mathcal{P}\left(x^{k+1} - \alpha_k \nabla \phi(x^k)\right)$$

Projected SGD:

$$x^{k+1} = \mathcal{P}\left(x^{k+1} - \alpha_k g(x^k)\right)$$
$$\mathbb{E}[g(x)] = \nabla \phi(x)$$

Converges, if slowly, for appropriate α_k (hard to choose).
And each step is cheap.

Coordinate Descent



(Block) Coordinate Descent

Idea: Partition variables and sweep over them; at each step

- Fix all but the current (block) of variables
- Take an optimization step on that variable (block)

Repeat until convergence.

Convergence[1] if each subproblem has a unique solution.
Q: What should the blocks / subproblems be?

[1]to a stationary point

Three Variants on Coordinate Descent

- Simple coordinate descent
    - Very cheap updates
    - Looks like PGD, but less parallel (GS vs Jacobi)
- $2k$ vector blocks (columns of W and rows of H)
    - Hierarchical Alternating Least Squares (HALS) / Rank-One Residual Iteration (RRI)
    - Subsolves are single-variable non-negative LS problems
    - Equivalent to simple coordinate descent (for one order)
- Alternately minimize on W and H
    - Alternating Non-Negative Least Squares (ANLS)
    - Yields one NNLS per row of W or column of H
    - Simple active-set solvers can work pretty well here!

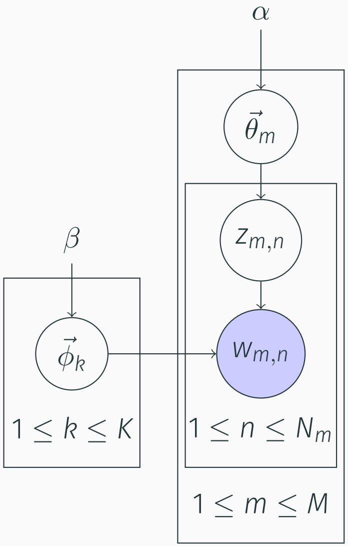
Can accelerate any of these (e.g. Anderson acceleration).
See example in Walker and Ni, SINUM 2011.

The Bayesian Approach: LDA

Latent Dirichlet Allocation (LDA) is a generative model:

- For each topic, choose word distribution $\vec{\phi}_k \sim \mathrm{Dir}(\beta)$
- For each doc, choose topic distribution $\vec{\theta}_m \sim \mathrm{Dir}(\alpha)$
- For word n in document m
  - Choose topic $z_{m,n} \sim \mathrm{Cat}(\vec{\theta}_m)$
  - Choose word $w_{m,n} \sim \mathrm{Cat}(\vec{\phi}_{z_{m,n}})$

The Bayesian Approach: LDA

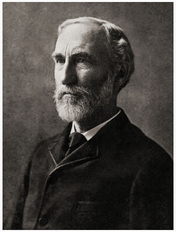


Graphical model notation:

- Circles = random variables
- Boxes = repetition
- Arrows = dependency
- Hyperparameters α, β

Observe $w_{m,n}$; infer the rest.

(Collapsed) Gibbs sampling for LDA



- Start with random assignment of topic labels $z_{m,n}$
- Compute counts (words per topic, topics per doc)
- Repeatedly loop over data; for each word $w_{m,n}$
    - Re-sample $z_{m,n}$ conditioned on all other $z_{m',n'}$
    - Update counts
- Infer $\vec{\phi}_k$ and $\vec{\theta}_m$ from samples of z

This looks a lot like coordinate descent!

The Picture So Far

- Lots of *locally* convergent iterations
- Slow convergence may be an issue?
    - Can still get useful results without guarantees!
    - Lee and Seung may not converge to stationary point
    - SGD methods are often terminated pretty early
    - Gibbs is useful because it sticks to a metastable state (otherwise permutation invariance blurs topic identity)
- All involve many passes over the data

One Stone, Two Birds

What makes us unhappy?

- Limited theory + frequent global convergence in practice
- Algorithms that require many passes over the data

A common solution: *separability*

Separable NMF

Separable structure:

$$\Pi^T A = \begin{bmatrix} I \\ W_2 \end{bmatrix} H$$

Happens in many domains!

Domain	Object ($a_{:,j}$)	Cluster ($w_{:,k}$)	Basis
Document	Word	Topic	Anchor word
Image	Pixel	Segment	Pure pixel
Network	User	Community	Representative
Legislature	Member	Party/Group	Partisan
Playlist	Song	Genre	Signature song

Finding Anchors

Assume separable structure:

$$\Pi^T A = \begin{bmatrix} I \\ W_2 \end{bmatrix} H$$

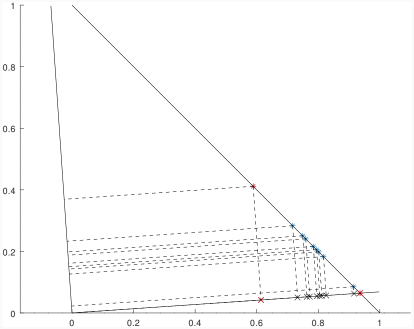
Normalize rows of A (rows of W, cols of H) to sum to 1:

$$\Pi^T \bar{A} = \begin{bmatrix} I \\ \bar{W}_2 \end{bmatrix} \bar{H}$$

Leading rows of $\Pi^T \bar{A}$ are the *convex hull* of all rows.

We have already seen how to find convex hulls!

Finding Anchors

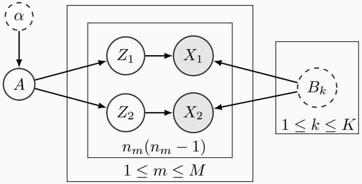


This is exactly the construction of ID via pivoted QR!

$$\bar{A}^T \Pi = QR = \bar{H}^T \begin{bmatrix} I & \bar{W}_2^T \end{bmatrix}$$

Recognizing pivoted QR lets us use better implementations.

Concentrate on Co-occurrence



Rest of the talk: focus on normalized co-occurrence matrix

$C_{ij} \propto$ frequency with which words i, j appear together

Note:

- Smaller (if less sparse) than word-document matrix
- Separability of word-document matrix inherited by C
- Topics tend to be easier to pull out
- Can produce a graphical model for C directly

Factoring the Co-occurrence

$$C \approx BAB^T$$

- Columns of B correspond to word distributions for topics
- Matrix A corresponds to topic correlation
- Entries of C are non-negative sum to 1 (joint stochastic)
- Entries of A are non-negative and sum to 1
- Columns of B are non-negative and sum to 1

The Anchor Words Algorithm

$$\Pi^T C \Pi \approx \begin{bmatrix} I \\ B_2 \end{bmatrix} A \begin{bmatrix} I & B_2^T \end{bmatrix}$$

- Find anchor words by pivoted QR on $\bar{C}$
- Simplex-constrained least squares to compute B_2
- Choose A as submatrix of C (can also do least squares)

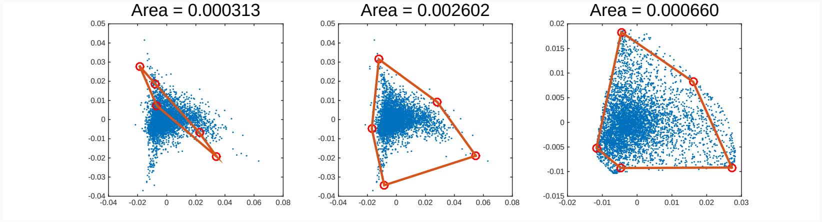
How Well Does It Work? (NeurIPS document collection)

Arora et al. 2013 (Baseline)
neuron layer hidden recognition signal cell noise
neuron layer hidden cell signal representation noise
neuron layer cell hidden signal noise dynamic
neuron layer cell hidden control signal noise
neuron layer hidden cell signal recognition noise
Probabilistic LDA (Gibbs)
neuron cell visual signal response field activity
control action policy optimal reinforcement dynamic robot
recognition image object feature word speech features
hidden net layer dynamic neuron recurrent noise
gaussian approximation matrix bound component variables

What Goes Wrong?

Not a conventional NeurIPS author.

What Goes Wrong?



- Naive anchor words algorithm finds bad anchors!
- Issue: Separability may only be approximate
- Data does not even look very low rank

Finding a Fix

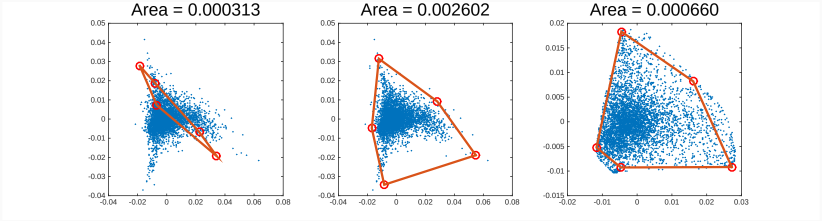
Model implicitly assumes C is

- Joint stochastic
- Positive semi-definite
- Low rank

Cannot fit outside this structure.

So: what if we enforce the structure?

Rectification



Alternating projection based on:

- Projection onto low-rank PSD (via eigensolve)
- Projection onto joint stochastic (elementwise)

Anchor selection based on rectified C works *much* better.

How Well Does It Work?

Lee et al. 2015 (AP)
neuron circuit cell synaptic signal layer activity
control action dynamic optimal policy controller reinforcement
recognition layer hidden word speech image net
cell field visual direction image motion object orientation
gaussian noise hidden approximation matrix bound examples
Probabilistic LDA (Gibbs)
neuron cell visual signal response field activity
control action policy optimal reinforcement dynamic robot
recognition image object feature word speech features
hidden net layer dynamic neuron recurrent noise
gaussian approximation matrix bound component variables

Summary

Three flavors of latent factor model:

- Latent factors via SVD are easy to compute
- Factors drawn from data are more interpretable
- NMF and LDA give natural parts decomposition

NMF is tricky in general, but

- Non-convexity sometimes seems a non-issue
- *Separability* hypothesis suggests why
- Assuming separability gives fast algorithms
- May need to enforce structure for good results

Recent approaches involve all three ideas: SVD/SEP, QRP, NMF.