Low-rank Matrix Factorization

SVD and beyond

Dmitry Adamskiy

Outline

Motivation

Motivation: matrix completion

- Quiz: what are the missing entries?

$$A = \begin{bmatrix} 7 & ? & ? \\ ? & 8 & ? \\ ? & 12 & 6 \\ ? & ? & 8 \\ 21 & 6 & ? \end{bmatrix}$$

Motivation: matrix completion

- Quiz: what are the missing entries?

$$A = \begin{bmatrix} 7 & ? & ? \\ ? & 8 & ? \\ ? & 12 & 6 \\ ? & ? & 8 \\ 21 & 6 & ? \end{bmatrix}$$

- Of course, the problem is ill-posed! They could be any numbers!

Motivation: matrix completion

- Quiz: what are the missing entries?

$$A = \begin{bmatrix} 7 & ? & ? \\ ? & 8 & ? \\ ? & 12 & 6 \\ ? & ? & 8 \\ 21 & 6 & ? \end{bmatrix}$$

- Of course, the problem is ill-posed! They could be any numbers!
- Suppose now that I tell you that the matrix is "nice". For example, it is rank-1. Could you reconstruct it now? What is A_{13}?

Motivation: matrix completion

- Quiz: what are the missing entries?

$$A = \begin{bmatrix} 7 & ? & ? \\ ? & 8 & ? \\ ? & 12 & 6 \\ ? & ? & 8 \\ 21 & 6 & ? \end{bmatrix}$$

- Of course, the problem is ill-posed! They could be any numbers!
- Suppose now that I tell you that the matrix is “nice”. For example, it is rank-1. Could you reconstruct it now? What is A_{13}?
- Of course, this is an extreme assumption. Relaxing it to low-rank approximation makes it a reasonable one.

Rank: a quick recap

Several equivalent definitions of matrix A being of rank k:

- The largest linearly independent set of columns has size k.
- The largest linearly independent set of rows has size k.
- Matrix A can be written as a sum of k rank-one matrices and cannot be written as a sum of $k-1$ rank-one matrices.
- If A is of size $(m \times n)$, it can be factored into the product of "long and skinny" $(m \times k)$ matrix Y and "short and wide" $(k \times n)$ matrix Z^T and cannot be factored into the product of $(m \times k-1)$ and $(k-1 \times n)$ matrices.
- ...

Rank: a quick recap

Several equivalent definitions of matrix A being of rank k:

- The largest linearly independent set of columns has size k.
- The largest linearly independent set of rows has size k.
- Matrix A can be written as a sum of k rank-one matrices and cannot be written as a sum of $k-1$ rank-one matrices.
- If A is of size $(m \times n)$, it can be factored into the product of "long and skinny" $(m \times k)$ matrix Y and "short and wide" $(k \times n)$ matrix Z^T and cannot be factored into the product of $(m \times k-1)$ and $(k-1 \times n)$ matrices.
- ...

Exercise: Show that these are all equivalent.

Rank: a quick recap

Several equivalent definitions of matrix A being of rank k:

- The largest linearly independent set of columns has size k.
- The largest linearly independent set of rows has size k.
- Matrix A can be written as a sum of k rank-one matrices and cannot be written as a sum of $k-1$ rank-one matrices.
- If A is of size $(m \times n)$, it can be factored into the product of "long and skinny" $(m \times k)$ matrix Y and "short and wide" $(k \times n)$ matrix Z^T and cannot be factored into the product of $(m \times k-1)$ and $(k-1 \times n)$ matrices.
- ...

Exercise: Show that these are all equivalent.

Quick quiz: Which rather large symmetric rank-one matrix have all of you seen?

Low-rank approximation:motivations

- Our goal is to find a "best" approximation for A of rank k:

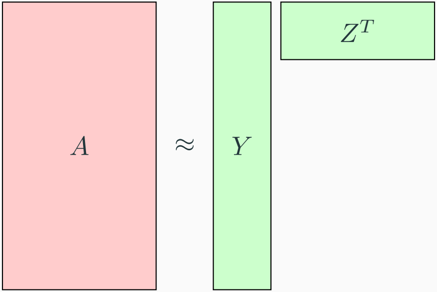


Figure 1: Low-rank approximation of matrix A

- Why would we like that?

Low-rank approximation:motivations

- Our goal is to find a "best" approximation for A of rank k:

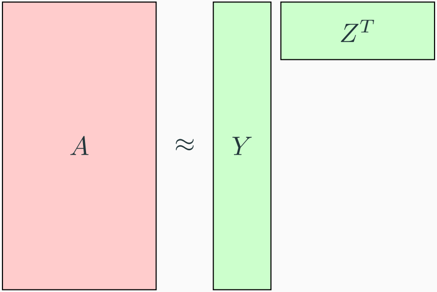


Figure 1: Low-rank approximation of matrix A

- Why would we like that?
    - Compression. $k(m+n)$ instead of mn numbers to store.

Low-rank approximation:motivations

- Our goal is to find a “best” approximation for A of rank k:

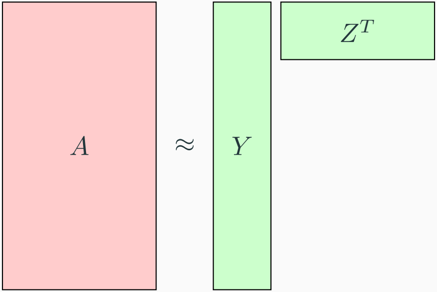


Figure 1: Low-rank approximation of matrix A

- Why would we like that?
  - Compression. $k(m+n)$ instead of mn numbers to store.
  - De-noising. If A is a noisy version of some low-rank signal, the reconstruction could actually be more informative. PCA.

Low-rank approximation:motivations

- Our goal is to find a "best" approximation for A of rank k:

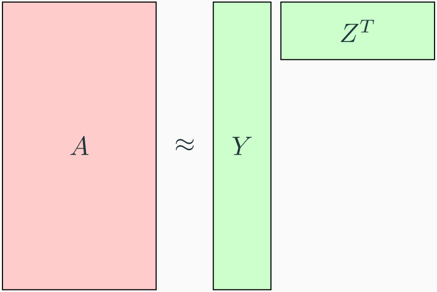


Figure 1: Low-rank approximation of matrix A

- Why would we like that?
  - Compression. $k(m+n)$ instead of mn numbers to store.
  - De-noising. If A is a noisy version of some low-rank signal, the reconstruction could actually be more informative. PCA.
  - Matrix completion.

Low-rank approximation:motivations

- Our goal is to find a "best" approximation for A of rank k:

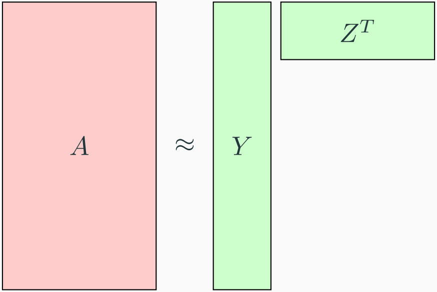


Figure 1: Low-rank approximation of matrix A

- Why would we like that?
  - Compression. $k(m+n)$ instead of mn numbers to store.
  - De-noising. If A is a noisy version of some low-rank signal, the reconstruction could actually be more informative. PCA.
  - Matrix completion.
  - ...

Motivation: Dimensionality reduction (PCA)

- PCA: probably the most popular technique for dimensionality reduction.
- Idea: find the linear subspace, such that projecting the data on this subspace maximises the variance (== minimizes the sum of squared distances from the data points to the projections).
- This could be done by a greedy algorithm!
- Equivalent to the truncated SVD.

Motivation: word vectors!

The traditional approach to word embeddings was to factorise word-word cooccurence matrix (with some pre- and post-processing).

- When word2vec appeared, people thought [2] that neural ‘predict’ methods outperform traditional distributional semantics ones.
- GloVe made the connection between the two clearer
- Levy at al. [4] think that there’s no overall winner and SVD performs on par with SGNS ang GloVe.

Word vectors: the idea

"You shall know a word by the company it keeps" (Firth, 1957)

- Use a sliding window of fixed size
- Compute word-word coocurences n_{uv}

Word vectors: the idea

"You shall know a word by the company it keeps" (Firth, 1957)

- Use a sliding window of fixed size
- Compute word-word coocurences n_{uv}
- Better: Pointwise Mutual Information

$$PMI = \log \frac{p(u,v)}{p(u)p(v)} = \log \frac{n_{uv}n}{n_u n_v}$$

Word vectors: the idea

"You shall know a word by the company it keeps" (Firth, 1957)

- Use a sliding window of fixed size
- Compute word-word coocurences n_{uv}
- Better: Pointwise Mutual Information

$$PMI = \log \frac{p(u,v)}{p(u)p(v)} = \log \frac{n_{uv}n}{n_u n_v}$$

- Even better: Positive PMI:

$$pPMI = \max(0, PMI)$$

- Then you factorize this matrix and get word embeddings.

Topic modeling (an overly-simplistic toy example)

Suppose that we have a term-by-document matrix which could be factorised:

$$\begin{bmatrix} 1 & 3 & 2 & 0 \\ 2 & 0 & 1 & 3 \\ 3 & 1 & 2 & 4 \end{bmatrix} = \underbrace{\begin{bmatrix} 1 & 3 \\ 2 & 0 \\ 3 & 1 \end{bmatrix}}_{Y} \cdot \underbrace{\begin{bmatrix} 1 & 0 & \frac{1}{2} & \frac{3}{2} \\ 0 & 1 & \frac{1}{2} & -\frac{1}{2} \end{bmatrix}}_{Z^T}$$

- We could think of columns of Y as topics (say, "computers" and "nuclear energy") and the columns of Z^T as the mixture coefficients for each document.
- Second word only occurs in computer-related documents (e.g. "x86")
- Third one is more likely to be seen in the "computer" documents (e.g. "operator") and the first one in the nuclear energy ones (e.g. "meltdown").

Motivation: NNMF for Audio

- Non-negative matrix factorization approach could be used to address the source separartion problem (and musical transcription).
- Lots of extensions, inluding neural network approaches.
- See examples here.

SVD

The idea

Informally, the recipe will be as follows:

1. Describe A as a sum of components ranked by their importance...

The idea

Informally, the recipe will be as follows:

1. Describe A as a sum of components ranked by their importance...
2. ...keep k most important components.

The idea

Informally, the recipe will be as follows:

1. Describe A as a sum of components ranked by their importance...
2. ...keep k most important components.

Singular Vector Decomposition (SVD) is a way of doing precisely that.

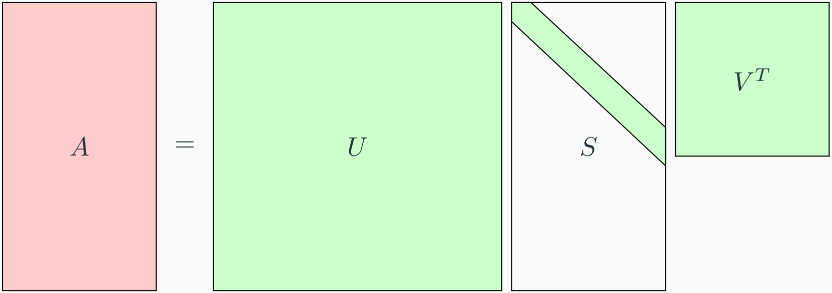


Figure 2: SVD decomposition of A

SVD

- SVD decomposes any $m \times n$ matrix A into $A = USV^T$, where
  1. U is an $m \times m$ orthogonal matrix (columns of U are called left singular vectors)
  2. V is an $n \times n$ orthogonal matrix (columns of V are called right singular vectors)
  3. S is an $m \times n$ diagonal matrix, with non-negative diagonal entries, sorted from high to low (these are called singular values).

This is equivalent to the following:

$$A = \sum_{i=1}^{\min(m,n)} s_i \cdot u_i v_i^T,$$

which is sum of $\min(m, n)$ rank-one matrices.

Properties of SVD

Every matrix has one and it is "more or less unique":

- The singular values are unique.
- When a singular value appears more than once, the subspaces spanned by corresponding vectors are unique, but the basis in each is arbitrary.

The running time is the smaller of $O(mn^2)$ and $O(m^2n)$. Computing k top singular values and corresponding vectors could be done faster.

Low-rank approximations from SVD

Following our recipe we restrict the sum to the k components with the largest singular values.

$$A \approx A_k = \sum_{i=1}^{k} s_i \cdot u_i v_i^T$$

Fact

This approximation is optimal in the following sense. For every rank-k matrix B,

$$||A - A_k||_F \leq ||A - B||_F,$$

where $||M||_F = \sqrt{\sum_{i,j} m_{i,j}^2}$ is Frobenius norm.

SVD intuitions and references

- Geomteric interpretation (rotation+scaling+rotation).
- I personally find that it is easiest to understand in the recommender system context.
- Jeremy Kun wrote a great blog post in two parts.

PCA and SVD

PCA reduces to SVD!

- PCA amounts to eigendecomposition of $A^T A$:

$$A^T A = QDQ^T$$

- This means that if $A = USV^T$, then

$$A^T A = VS^T U^T USV^T = VDV^T,$$

  where D is a diagonal matrix with diagonals equal to the squares of diagonal entries of S.
- This means that the eigenvectors of $A^T A$ (principal dimensions) are the same as right singular vectors of A.
- However, SVD also gives us matrix U. In some applications, we might be interested in it as well (collaborative filtering: one set of singular vectors is "canonical products", the other is "canonical customers").

How to compute SVD?

- Power method – conceptualy simple way to compute the eigenvector corresponding to the largest eigenvalue.
- Then we can subtract this component and recurse.
- Convergence is determined by the ratio σ_1/σ_2
- Industry strength methods are more involved.

Drawbacks

SVD is optimal in a sense described above, but for, say, topic modeling it has two drawbacks:

Drawbacks

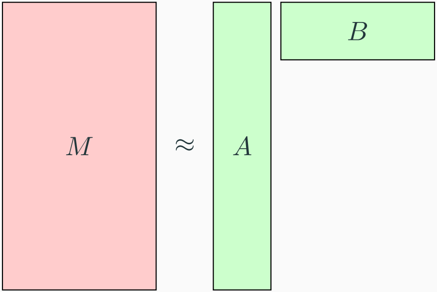
SVD is optimal in a sense described above, but for, say, topic modeling it has two drawbacks:

- The orthogonality of singular vectors. Topics like "basketball" and "football" are distinct, but similar.
- The negative entries (bad for interpretability, especially probabilistic interpretation).

(N)NMF

This bring us to the idea of non-negative matrix factorisation (NMF or NNMF):

- For a matrix M or size $m \times n$ the goal is to express M, at least approximately as a product of non-negative $m \times k$ and $k \times n$ matrices A and B.
- Same as before, only now the factors are non-negative:



NMF: problem setting

The NMF problem is this:

$$\min ||X - WH||_F^2, W \geq 0, H \geq 0$$

(other loss functions are possible).

Issues:

- NP-hard in general
    - Use standard non-linear optimisation techniques
- Ill-posed. The solution is non-unique (and not just trivially up to permutations).
    - Priors, regularisation
    - leads to lots of variants of NMF
- How to choose factorisation rank
    - Trial and error

Standard NMF framework

- The standard framework is to solve problems in W and H iteratively.
  1. Fix H, optimize for W
  2. Fix W, optimize for H

Standard NMF framework

- The standard framework is to solve problems in W and H iteratively.
  1. Fix H, optimize for W
  2. Fix W, optimize for H
- Reason: the sub-problems are convex (NNLS)

Standard NMF framework

- The standard framework is to solve problems in W and H iteratively.
  1. Fix H, optimize for W
  2. Fix W, optimize for H
- Reason: the sub-problems are convex (NNLS)
- In fact, the subproblems are independent in, say, rows of W:

$$||X - WH||_F^2 = \sum_{i=1}^{k} W_{i:}(HH^T)W_{i:}^T - 2W_{i:}(HX_{i:}^T) + ||X_{i:}||_2^2$$

Optimality conditions and Multiplicative updates

First-order optimality conditions for NMF:

$$W \geq 0, \nabla_W F = WHH^Y - XH^T \geq 0; W \circ \nabla_W F = 0$$
$$H \geq 0, \nabla_H F = W^T WH - W^T X \geq 0; H \circ \nabla_H F = 0$$

The simple iterative algorithm called Multiplicative Updates is proposed in [3]:

$$W = W \circ \frac{[XH^T]}{[WHH^T]},$$

where $\frac{[]}{[]}$ is element-wise division and $\circ$ is element-wise multiplication.

Properties of MU

- Based on majorisation-minimisation framework (minimising the upper bound).
- Could be views as rescaled gradient method
- For each entry of W

$$\frac{[XH^T]_{ik}}{[WHH^T]_{ik}} \geq 1 \iff (\nabla_W F)_{ik} \leq 0)$$

Properties of MU

- Based on majorisation-minimisation framework (minimising the upper bound).
- Could be views as rescaled gradient method
- For each entry of W

$$\frac{[XH^T]_{ik}}{[WHH^T]_{ik}} \geq 1 \iff (\nabla_W F)_{ik} \leq 0)$$

- Is not guranteed to converge to a stationary point without tricks (intuitive reason – once $W_{ij} = 0$ it is never updated but its partial derivative may become negative). Several ways to fix that.

Alternating Least Squares

Another simple algorithm is Alternating Least Squares(ALS).

- Solve the unconstrained least-squares subproblem
- Then project. Simple (pseudo)code:

$$W = \max(0, (XH^T)/(HH^T))$$

- Usually does not converge, but good for initialisation purposes (run a few iterations before switching to something else).

Alternating Non-Negative Least Squares

This is a class of methods where the sub-problems are solved exactly.

$$W = \arg\min_{W \geq 0} ||X - WH||_F.$$

- Lots of methods to solve NNLS (active-set methods, projected gradients, Quasi-Newton.
- There is an enhanced version (HALS) that converges faster.

How to start and when to stop

- Various strategies proposed for the initialisation of W and H (for example. using SVD).
- No theoretical guarantees for any of them.
- Trial-and-error (use a few different ones and keep the best solution).
- Stopping critera include monitoring the objective function, optimality conditions and the difference between successive iterates.

NMF: Separable case

- NMF problem is NP-hard in the worst case.
- People use various approximate methods , converging to local minima.
- However, there is an assumption which makes it tractable [1].

Separability or Anchor Words assumption

Suppose that matrix M admits factorisation $M = AB$ such that for each column j of A, there exists a row i, such that $A_{ij} > 0$ and $A_{ik} = 0$ for $k \neq j$.

NMF: Separable case

- NMF problem is NP-hard in the worst case.
- People use various approximate methods , converging to local minima.
- However, there is an assumption which makes it tractable [1].

Separability or Anchor Words assumption

Suppose that matrix M admits factorisation $M = AB$ such that for each column j of A, there exists a row i, such that $A_{ij} > 0$ and $A_{ik} = 0$ for $k \neq j$.

Interpretation: for each topic there is a word which occurs only in the documents on this topic.

Algorithm for the exact case: Normalisation

We could simplify our problem a bit.

- Without the loss of generailty, $\sum_j M_{ij} = 1$.

Algorithm for the exact case: Normalisation

We could simplify our problem a bit.

- Without the loss of generailty, $\sum_j M_{ij} = 1$. (otherwise we normalise, factorise and renormalise)

Algorithm for the exact case: Normalisation

We could simplify our problem a bit.

- Without the loss of generarilty, $\sum_j M_{ij} = 1$. (otherwise we normalise, factorise and renormalise)
- Without the loss of generality, $\sum_j B_{ij} = 1$.

Algorithm for the exact case: Normalisation

We could simplify our problem a bit.

- Without the loss of generarilty, $\sum_j M_{ij} = 1$. (otherwise we normalise, factorise and renormalise)
- Without the loss of generality, $\sum_j B_{ij} = 1$. (otherwise we normalise the rows of B and compensate for is by renormalising the columns of A)

Algorithm for the exact case: Normalisation

We could simplify our problem a bit.

- Without the loss of generarilty, $\sum_j M_{ij} = 1$. (otherwise we normalise, factorise and renormalise)
- Without the loss of generality, $\sum_j B_{ij} = 1$. (otherwise we normalise the rows of B and compensate for is by renormalising the columns of A)
- But then $\sum_j A_{ij} = 1$ as well!

Algorithm for the exact case: Normalisation

We could simplify our problem a bit.

- Without the loss of generailty, $\sum_j M_{ij} = 1$. (otherwise we normalise, factorise and renormalise)
- Without the loss of generality, $\sum_j B_{ij} = 1$. (otherwise we normalise the rows of B and compensate for is by renormalising the columns of A)
- But then $\sum_j A_{ij} = 1$ as well! The reason is this: rows of M are non-negative linear combinations of rows of B with coefficients in the rows of A. Since rows M and B sum to 1, this has to be convex combination!

Algorithm for the exact case: Normalisation

We could simplify our problem a bit.

- Without the loss of generailty, $\sum_j M_{ij} = 1$. (otherwise we normalise, factorise and renormalise)
- Without the loss of generality, $\sum_j B_{ij} = 1$. (otherwise we normalise the rows of B and compensate for is by renormalising the columns of A)
- But then $\sum_j A_{ij} = 1$ as well! The reason is this: rows of M are non-negative linear combinations of rows of B with coefficients in the rows of A. Since rows M and B sum to 1, this has to be convex combination!

Which means that we are looking for a matrix B, such that all rows of M in the convex hull of the rows of B.

Algorithm for NMF with anchor words

- Anchor words assumption means that there is identity matrix embedded in A.

Algorithm for NMF with anchor words

- Anchor words assumption means that there is identity matrix embedded in A.
- But that means that there are k rows of M embedded in B!

Algorithm for NMF with anchor words

- Anchor words assumption means that there is identity matrix embedded in A.
- But that means that there are k rows of M embedded in B!
- The goal is now to find k rows of M such that the rest of them are in the convex hull of these k.
- This could be done greedily. Start with all the rows of M. While there is a row that is in the convex hull of the rest: delete it.
- And this could be done with linear programming.

In real-life the separability assumption only holds approximately, and the we have to prove that the algorithms are robust to noise (Successive Projection Algorithm).

References I

S. Arora, R. Ge, and A. Moitra.
Learning topic models – going beyond svd.
In *Proceedings of the 2012 IEEE 53rd Annual Symposium on Foundations of Computer Science*, FOCS '12, pages 1–10, Washington, DC, USA, 2012. IEEE Computer Society.

M. Baroni, G. Dinu, and G. Kruszewski.
Don't count, predict! a systematic comparison of context-counting vs. context-predicting semantic vectors.
In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 238–247, Baltimore, Maryland, June 2014. Association for Computational Linguistics.

References II

D. D. Lee and H. S. Seung.
Algorithms for non-negative matrix factorization.
In T. K. Leen, T. G. Dietterich, and V. Tresp, editors, *Advances in Neural Information Processing Systems 13*, pages 556–562. MIT Press, 2001.

O. Levy, Y. Goldberg, and I. Dagan.
Improving distributional similarity with lessons learned from word embeddings.
TACL, 3:211–225, 2015.