Large-scale data analysis via matrix and tensor decompositions

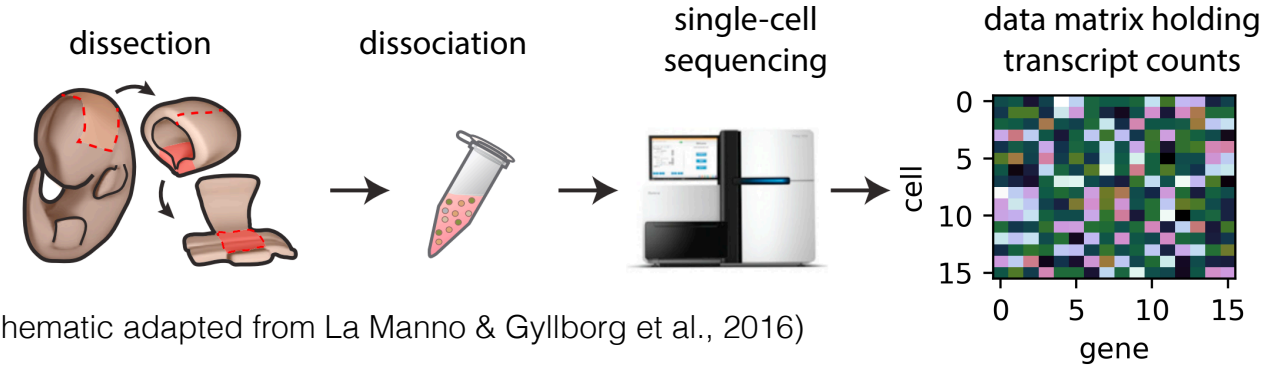
Part 1: Matrix decomposition

Alex Williams

MIT, 09/05/2017

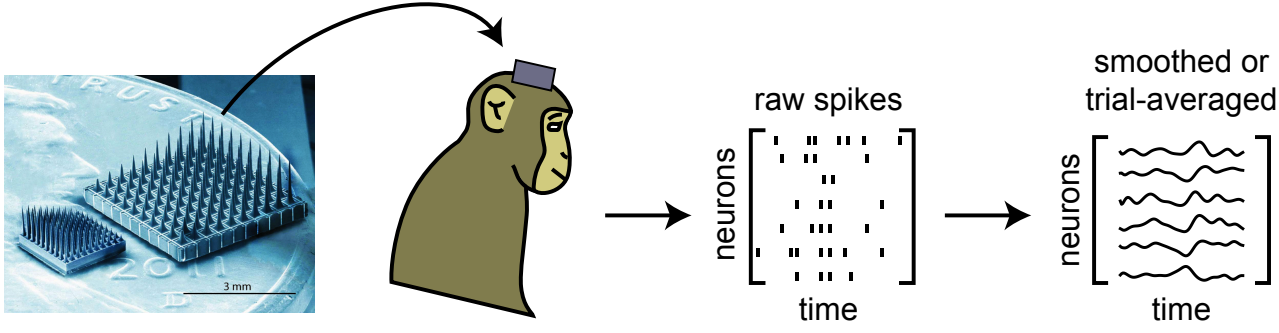
Examples of Matrix-Encoded Data

1. Gene Expression



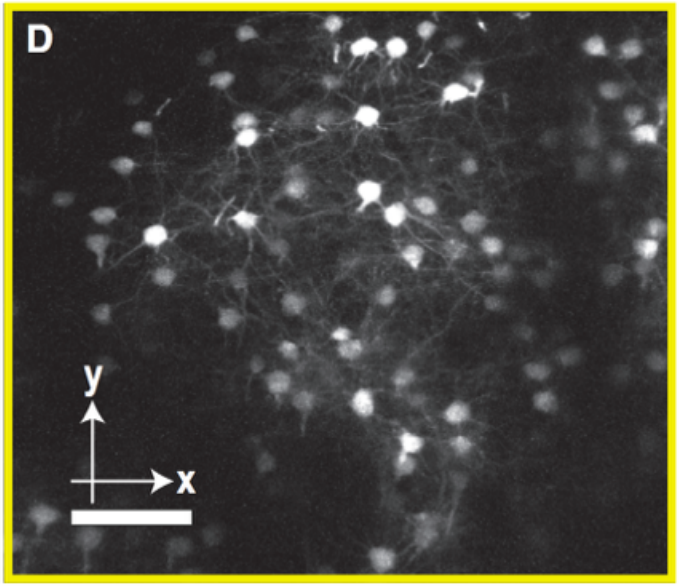
(schematic adapted from La Manno & Gyllborg et al., 2016)

2. Neural Activity



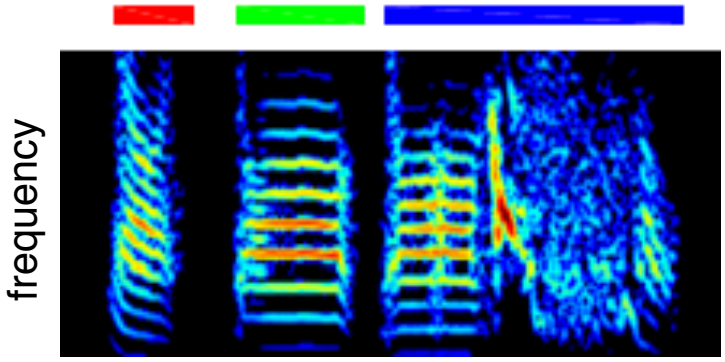
Examples of Matrix-Encoded Data

3. Fluorescence Images



Cortical neurons expressing YFP
(Kim & Zhang et al., 2016)

4. Spectrograms

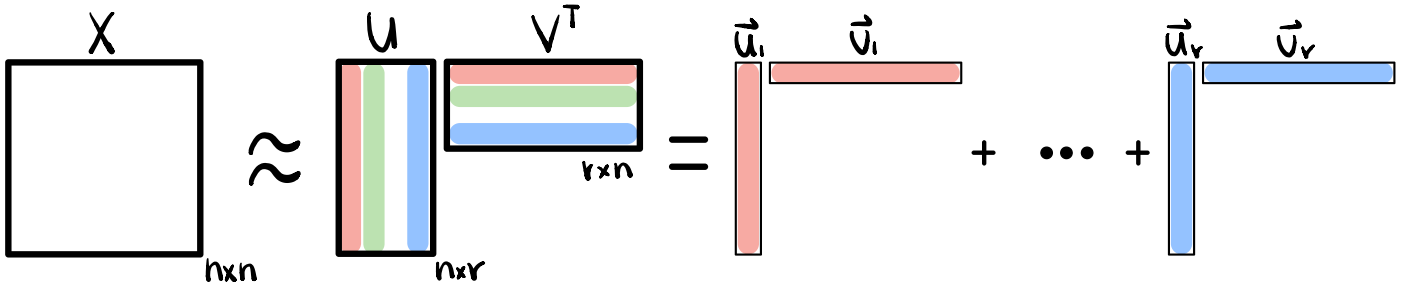


Zebra Finch courtship song
(Provided by Emily Mackevicius)

1. 矩阵分解的通用架构

1.1 从数学角度来理解矩阵分解

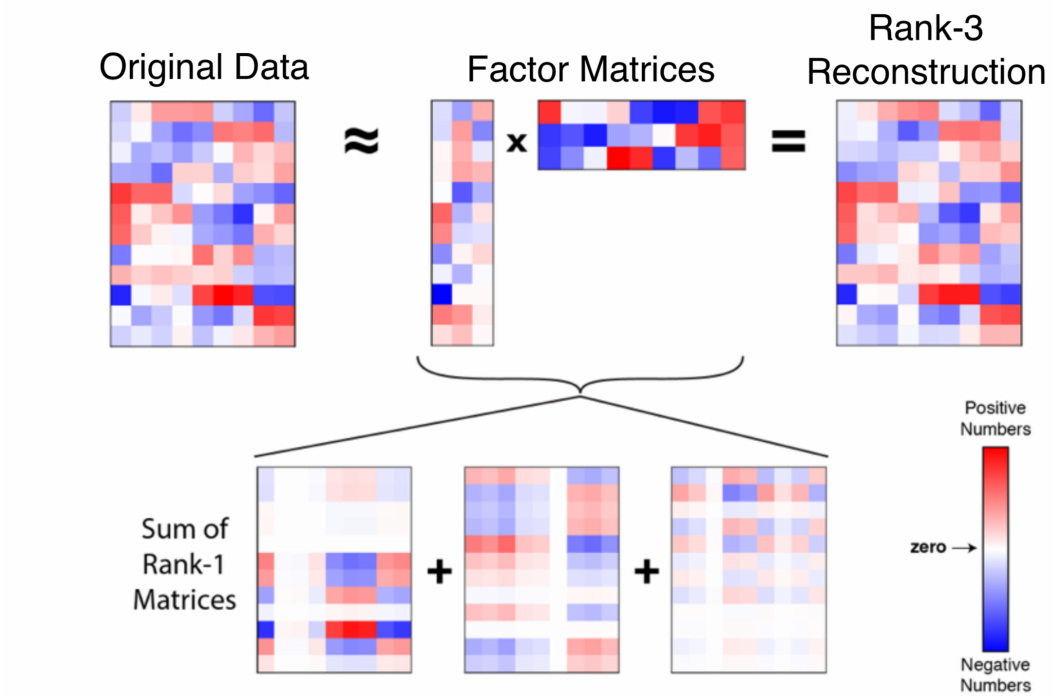
对于一个 $n \times n$ 的矩阵 X，可以用 tall skinny matrix U 和 shot fat matrix V 来估计。



$$\begin{matrix} x_{11}, x_{12}, \cdots, x_{1n} \\ x_{21}, x_{22}, \cdots, x_{2n} \\ \vdots \quad \vdots \quad \ddots \quad \vdots \\ x_{n1}, x_{n2}, \cdots, x_{nn} \end{matrix} \approx \begin{matrix} u_{11}v_{11}, u_{11}v_{21}, \cdots, u_{11}v_{r1} \\ u_{21}v_{11}, u_{21}v_{21}, \cdots, u_{21}v_{r1} \\ \vdots \quad \vdots \quad \ddots \quad \vdots \\ u_{n1}v_{11}, u_{n1}v_{21}, \cdots, u_{n1}v_{r1} \end{matrix} + \cdots + \begin{matrix} u_{1r}v_{1r}, u_{1r}v_{2r}, \cdots, u_{1r}v_{nr} \\ u_{2r}v_{1r}, u_{2r}v_{2r}, \cdots, u_{2r}v_{nr} \\ \vdots \quad \vdots \quad \ddots \quad \vdots \\ u_{nr}v_{1r}, u_{nr}v_{2r}, \cdots, u_{nr}v_{nr} \end{matrix}$$

$$x_{ij} \approx \sum_{r=1}^{r} u_{ir} \cdot v_{rj}$$

例如：



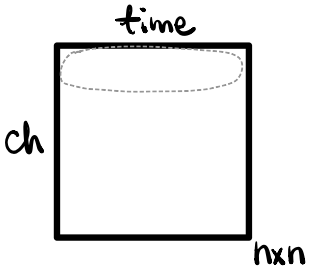
通过矩阵分解，我们就可以把一个结构复杂的矩阵 X 表示为 $r=3$ 个秩为1的结构简单矩阵的和。

The definition of a rank one matrix is that every column is just a scalar multiple of the other columns and likewise every row is a scalar multiple of the other rows.
Each one of these rank one matrix is actually a very simple low dimensional pattern and but even just sort of summing together three of them, you get a decent amount of structure.

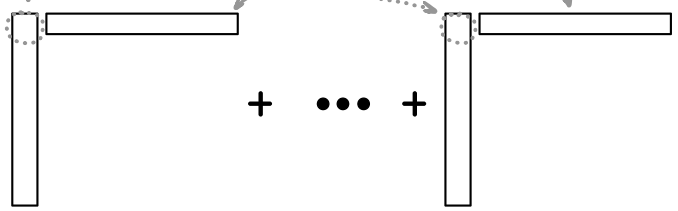
1.2 从时空生理数据角度来理解矩阵分解

对于实验数据 X

若考察对象是每个 channel 的信号

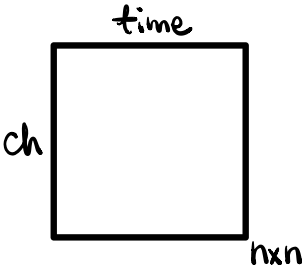


≈

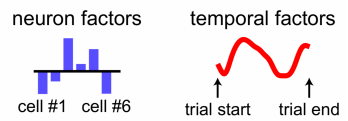


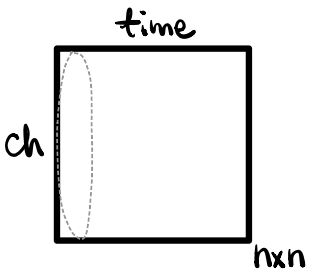
则将 U_j $(j=1\sim r)$ 理解为组成一个 channel 1 信号的多 component，

U_{1j} $(j=1\sim r)$ 为 channel 1 具有的多 temporal components 的“量”。

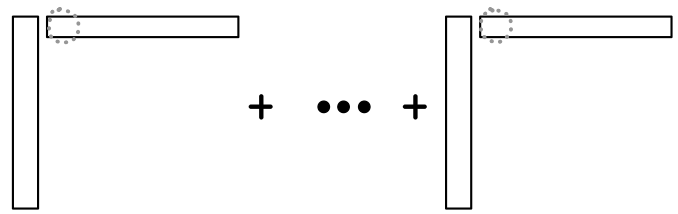


若考察对象是所有 channel 在一个时间点的信号





≈



综上，任何矩阵分解（PCA，NMF，…）都基于此架构，

只不过优化 U，V 的 loss func（+ regularization + constrain）不同，

探一个角度来讲，由于不同实验数据本身具有不同的特性（非负，稀疏，幅度较小，0-1，outliers，…），我们对 components 的要求也不同，采用的矩阵分解方法就不同，U、V 优化的效果也不同。

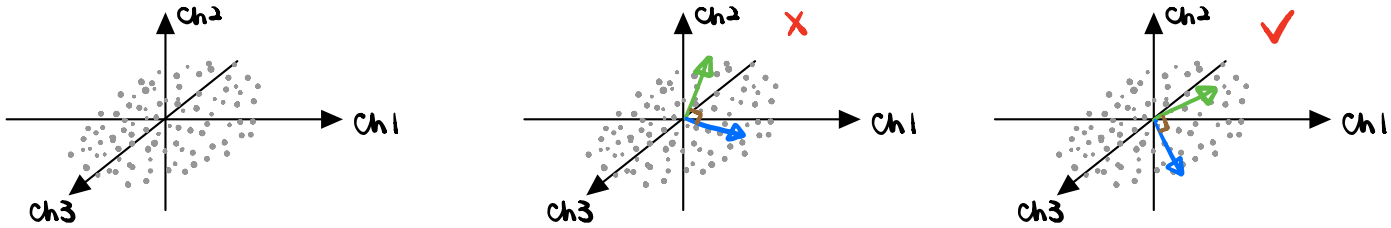
2. PCA

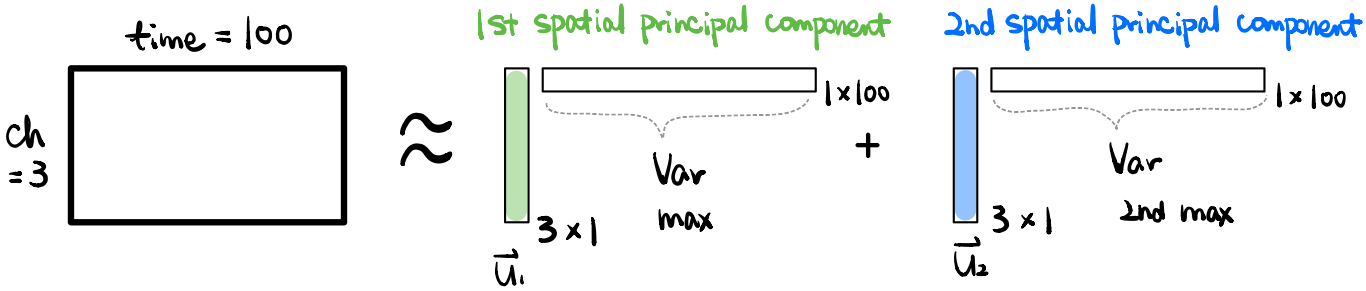
①每个channel去均值

②考察对象是所有channel在一个时间点的信号，即每一列为三维空间中的一个点。

③这100个点关于原点对称

我们希望利用矩阵分解，在三维空间中找到 $r<3$，例如 $r=2$ 个规范正交基（spatial components），使得这些点在 1st 基的投影方差最大，而在 2nd 基的投影方差第二大。





原始数据点在 principal component 上的投影方差越大，说明原始数据点的多样性（信息）被保留在了投影中。

若一组 spatial principal component 能（使投影）最大程度地保留原始数据的方差 $\|X\|_F^2$，则说明其最能代表 spatial pattern at each time point.

即，在每个时间点的原始数据可由这组 spatial principal component 线性估计。

因此，我们的优化目标为：

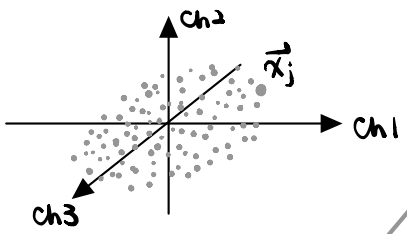
$$\max_{U} \quad X^T \cdot U \cdot (X^T \cdot U)^T$$
$$\text{subject to} \quad U^T \cdot U = I$$

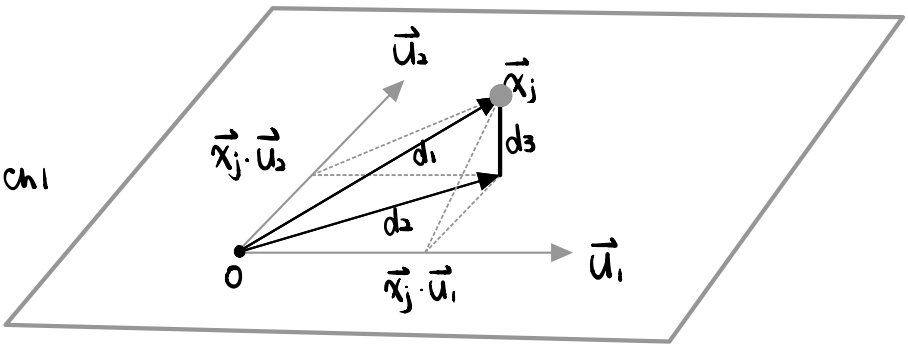
PCA 的最大可分性

从几何角度来理解最大可分性：

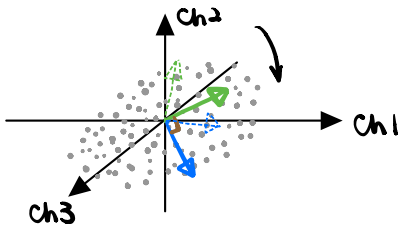
我们希望在三维空间中，找到一个由规范正交基 $\vec{u}_1$，$\vec{u}_2$ 组成的二维嵌套平面。每一个点 $\vec{x}_j$ 的方差为 $\|\vec{x}_j\|_F^2 = d_1^2$，由两个 spatial components $\vec{u}_1$ $\vec{u}_2$ 所表示的部分方差为 $(\vec{x}_j \cdot \vec{u}_1)^2 + (\vec{x}_j \cdot \vec{u}_2)^2 = d_2^2$。

因为 $d_1^2 = d_2^2 + d_3^2$，且 $d_1^2 = \text{const}$，所以 $\max d_2^2 = \min d_3^2$

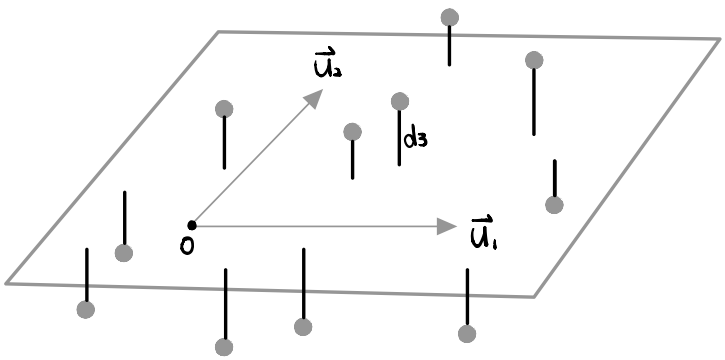




所以，优化 spatial components $\vec{u}_1$ $\vec{u}_2$ 使其最大程度地保留原始数据 X 的方差



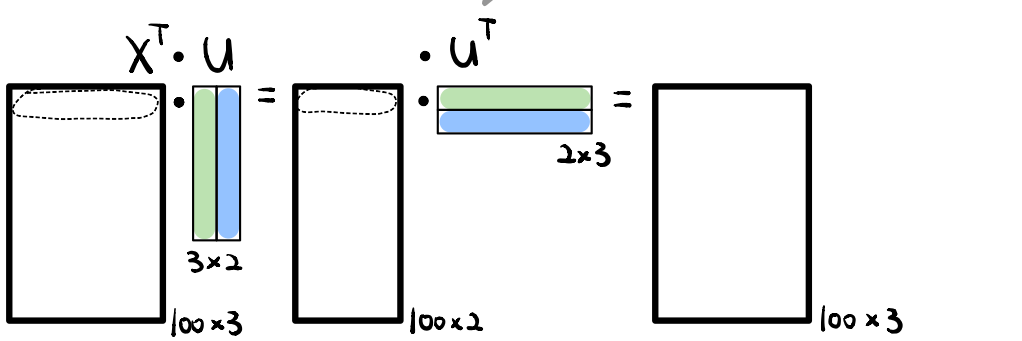
等价于在三维空间中，找到一个由规范正交基 $\vec{u}_1$ $\vec{u}_2$ 所打展成的低维嵌入平面，数据点向该平面的投影距离最小



因此，我们的优化目标又可以写为：

$$\min_{U} \ \| X^T - X^T \cdot U \cdot U^T \|_F^2$$
$$\text{subject to} \quad U^T \cdot U = I$$

最近重构性



综上，PCA 的优化目标（loss func）

最大可分性：

$$\max_U \quad X^T \cdot U \cdot (X^T \cdot U)^T$$

subject to $U^T \cdot U = I$

等价

最近重构性：

$$\min_U \quad \| X^T - X^T \cdot U \cdot U^T \|_F^2$$

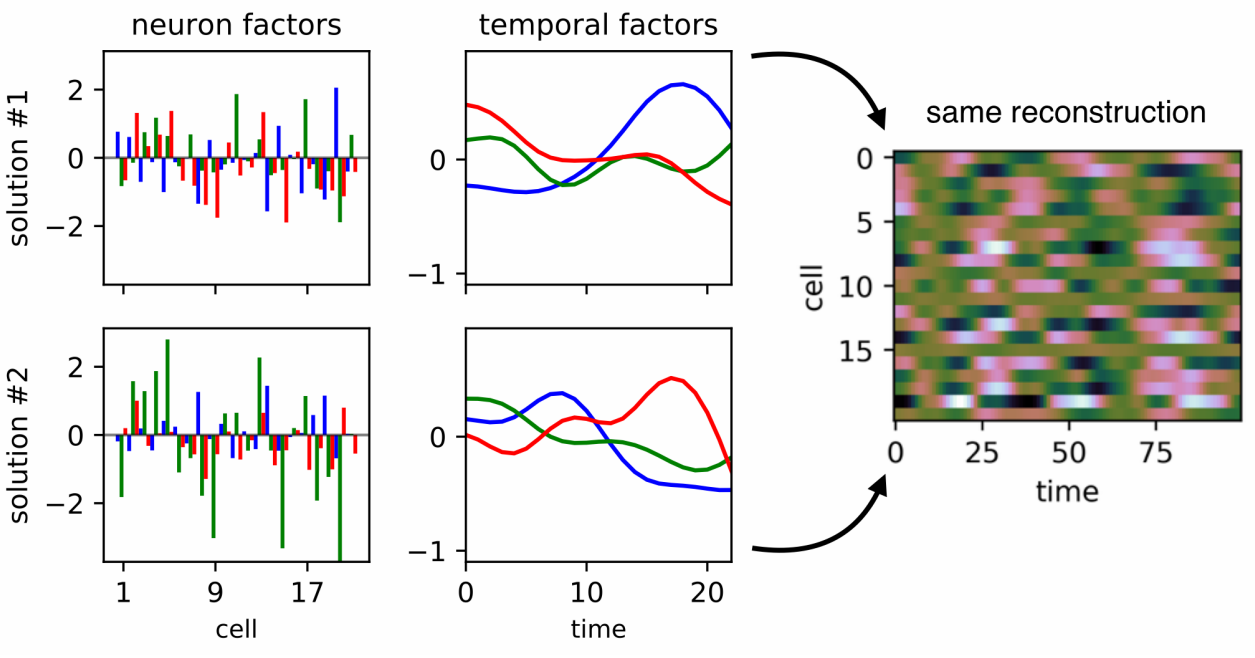
subject to $U^T \cdot U = I$

优化得到的 principle components 为原始数据中的特征 patterns，这些 patterns 最大程度地保留了原始数据中的信息，同时又降低了原始数据的维度。

There are an infinite # of solutions to PCA

known as "the rotation problem"

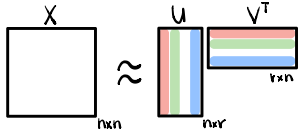
$$\hat{\mathbf{X}} = \mathbf{U}\mathbf{V}^T = \mathbf{U}\mathbf{F}^{-1}\mathbf{F}\mathbf{V}^T = \mathbf{U}'\mathbf{V}'^T$$

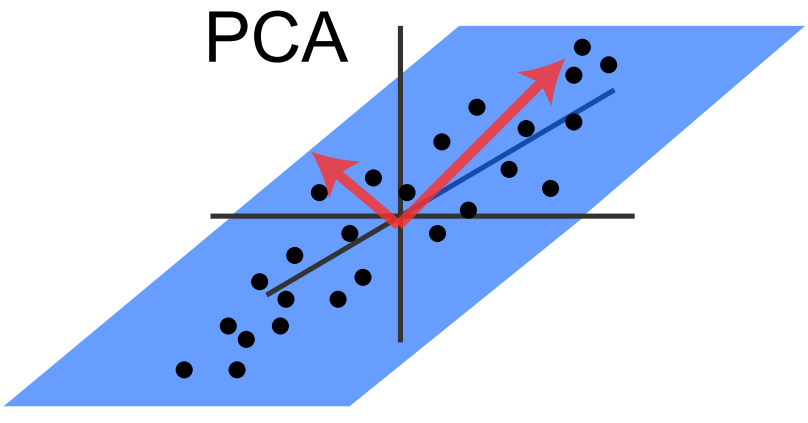


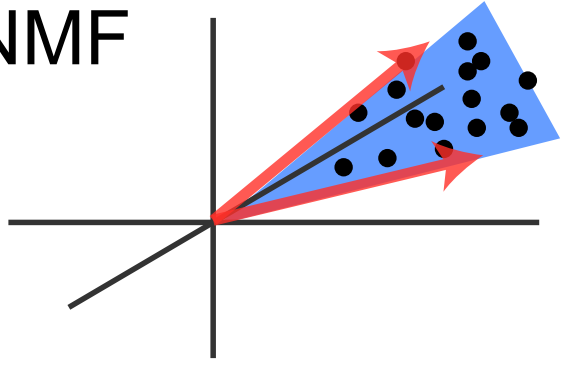
3. Nonnegative Matrix Factorization (NMF)

$$\underset{\mathbf{U},\mathbf{V}}{\text{minimize}} \quad \|\mathbf{X} - \mathbf{U}\mathbf{V}^T\|_F^2$$

$$\text{subject to} \quad \mathbf{U} \geq 0,\ \mathbf{V} \geq 0$$



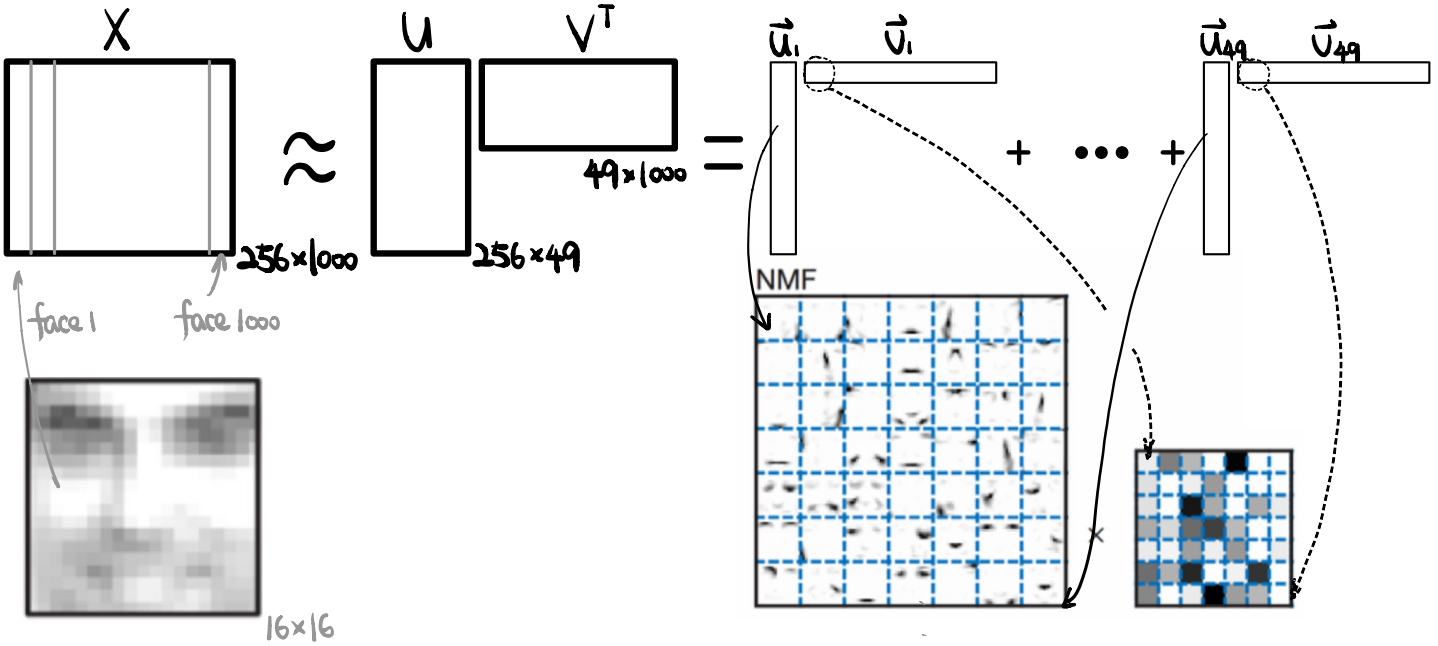




Sometime, your data only exist in the positive space, and all your data points must lie within a **cone**. So, in this case, it is a two-dimensional cone, and **the edges of the cone are defined by these basis vectors,** which could, depends on how you think about it, **be the rows of V, or the columns of U.**

To find these vectors, all your data points must lie in between these two vectors. And **if your data points are spread out far enough, then to reconstruct the data, the vector will basically be pushed to the very edges of the positive.** Intuitively you cannot rotate these vectors anymore and reconstruct the same data.

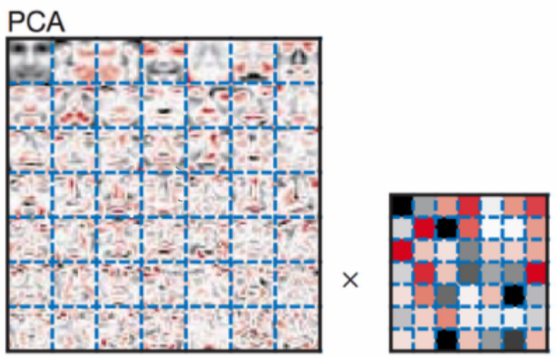
PCAとNMFの比較：



NMF advantages:

- sparse factors
- additively combined
- can be "parts-based"
- can be unique (i.e. no rotation problem)

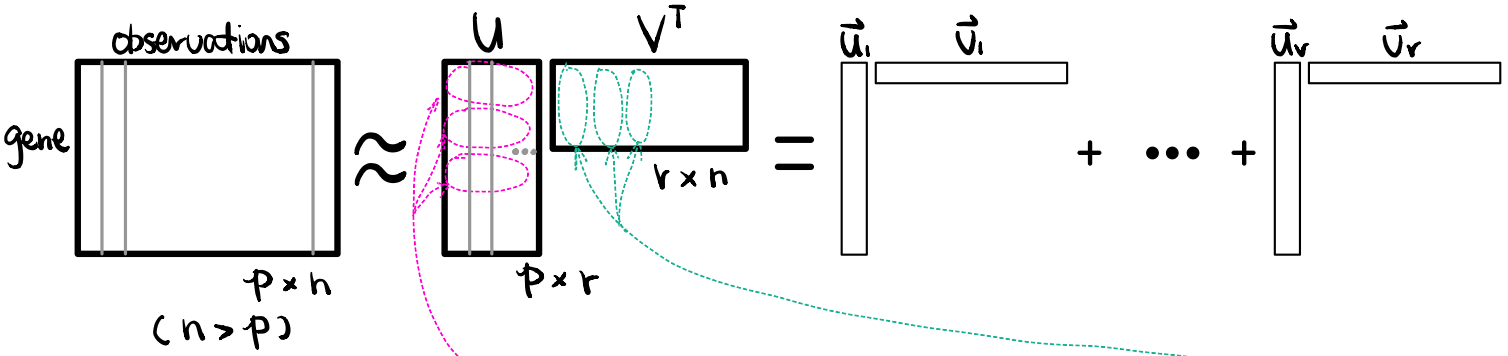
(Stodden & Donoho, 1999)



(Lee & Seung, 1999)

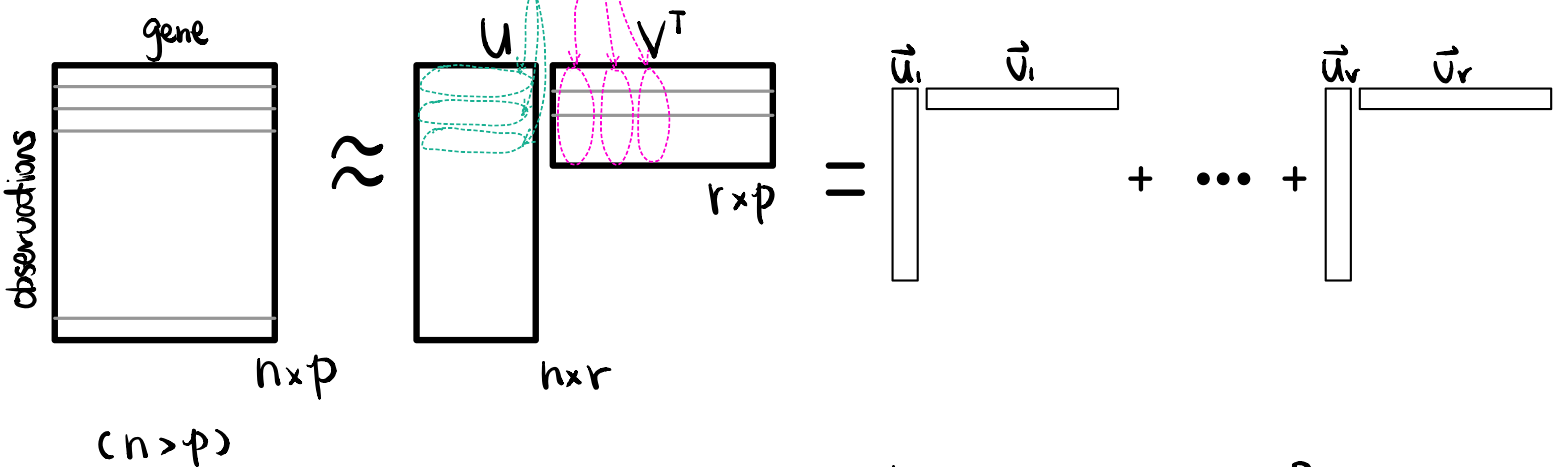
4. Sparse PCA

$$\underset{\mathbf{U},\mathbf{V}}{\text{minimize}} \quad \|\mathbf{X}-\mathbf{U}\mathbf{V}^T\|_F^2+\lambda_u\sum_{i=1}^{p}\|\mathbf{u}_{i:}\|_1+\lambda_v\sum_{j=1}^{n}\|\mathbf{v}_{j:}\|_2^2$$

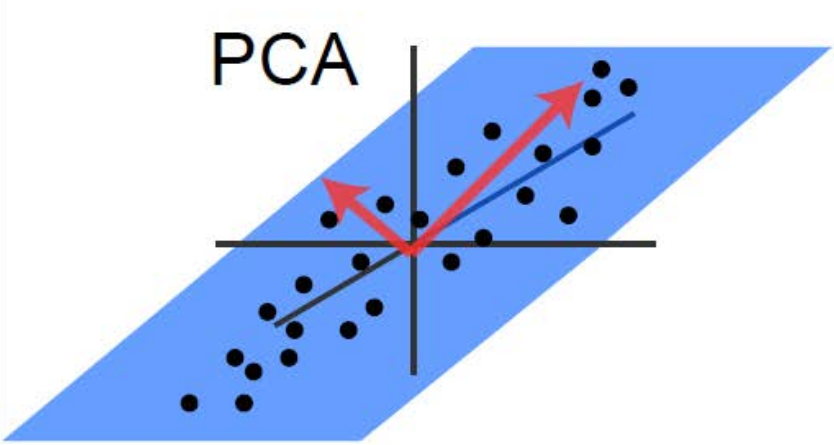


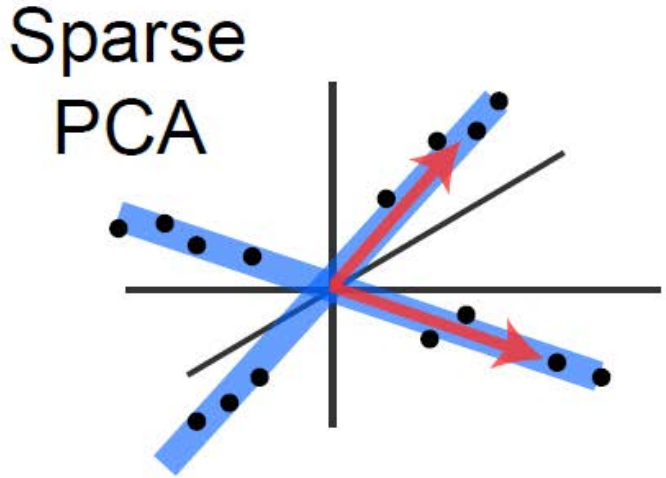
多 principle components 的每个"特征"保持稀疏. e.g., U的第一行(0, 2, 0, 1, 0, ···, 0)。也就是说. 多PCs仅对应原数据中少量基因特征。sparse pattern.

一个observation在多PCs的投影保持稀疏(L1) or 不能太大(L2)



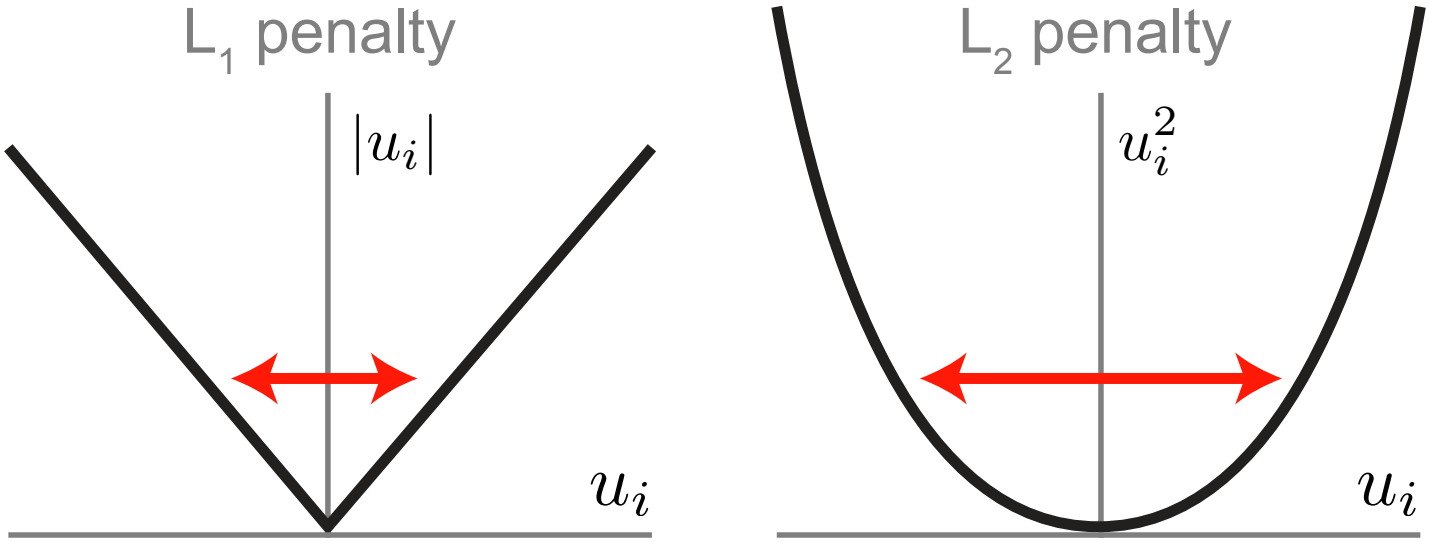
$$\underset{\mathbf{U},\mathbf{V}}{\text{minimize}} \quad \|\mathbf{X}-\mathbf{U}\mathbf{V}^T\|_F^2+\lambda_u\sum_{i=1}^{n}\|\mathbf{u}_{i:}\|_2^2+\lambda_v\sum_{j=1}^{p}\|\mathbf{v}_{j:}\|_1$$



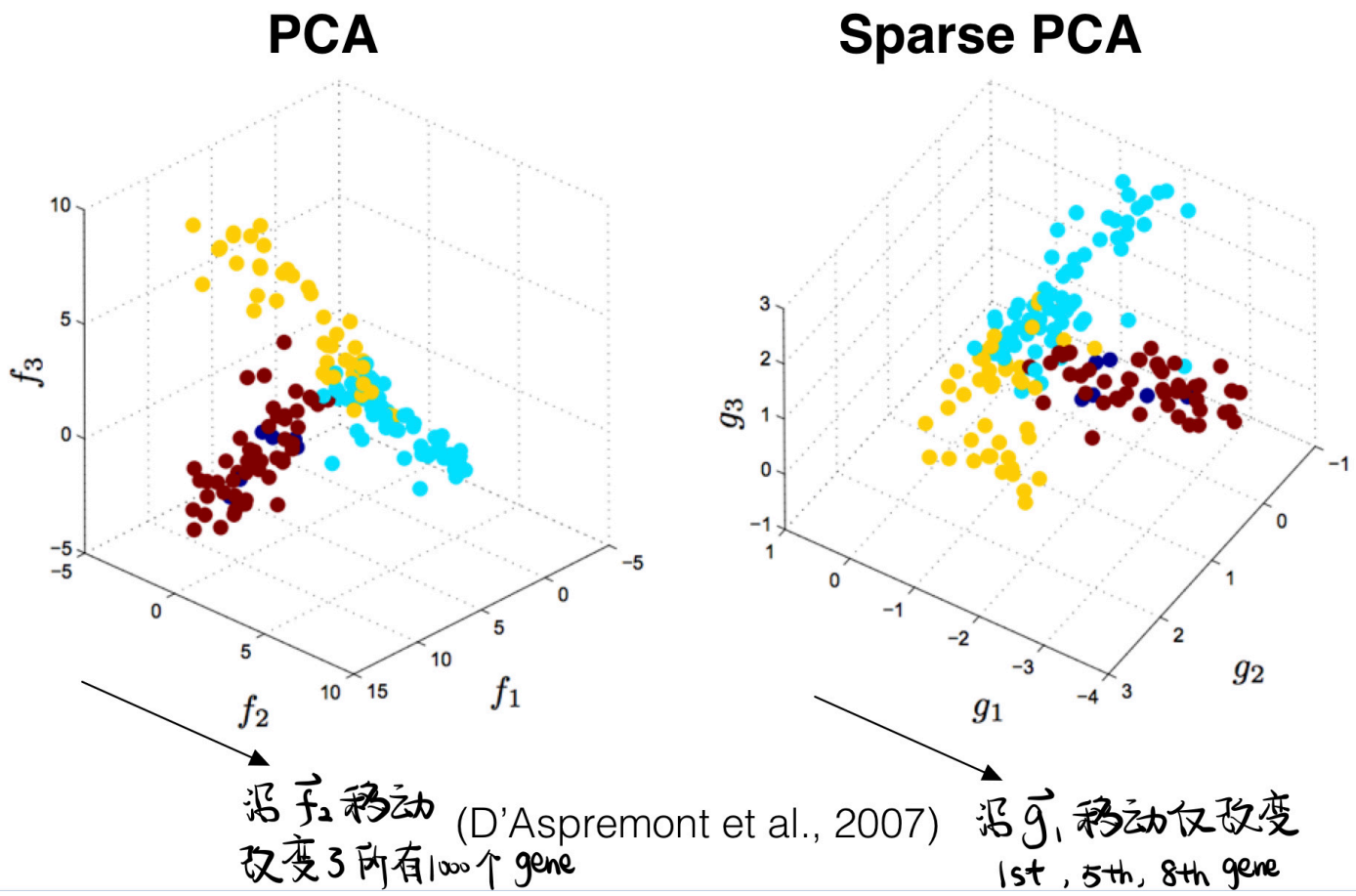


* Several variants of this model with different properties appear in the literature. Originally it was proposed by Zou et al. (2006).

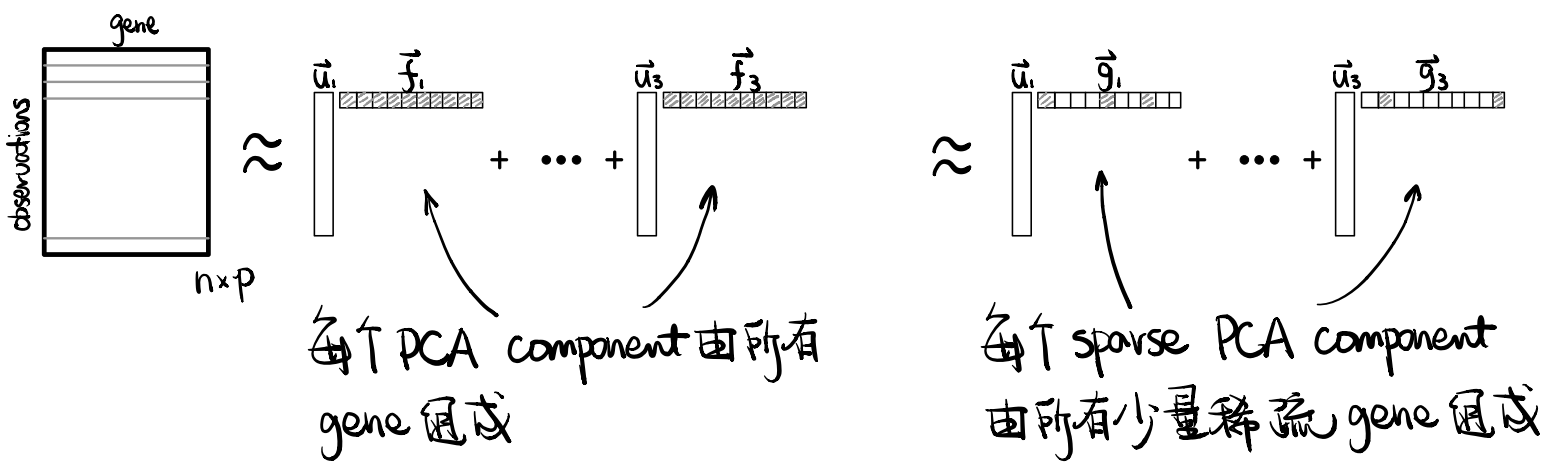
Why L1 penalties result in sparse factors



Sparse PCA



(D'Aspremont et al., 2007)

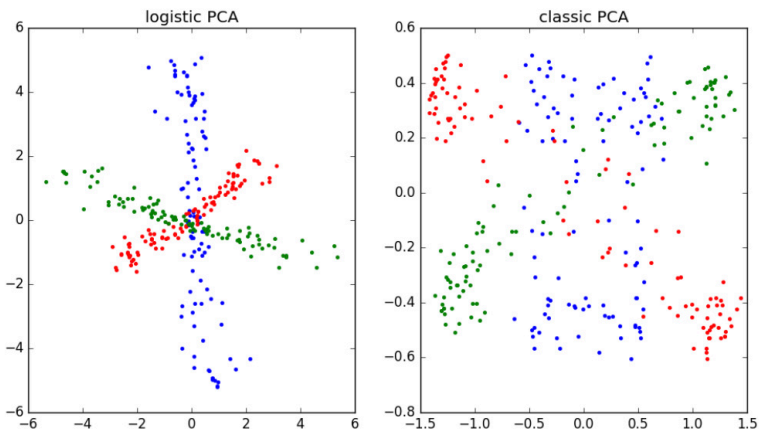


5. 其它 PCA 方法

Logistic PCA. We can also replace squared error with different loss functions. For example, suppose you sequence the genomes of n patients and check p nucleotide sites for mutations ($x_{ij} = 1$ if patient i has a mutation at site j, and $x_{ij} = -1$ if there is not mutation). Because your data is *binary* you might use logistic PCA, which is similar to logistic regression:

$$\underset{W,C}{\text{minimize}} \quad \sum_{i=1}^{n}\sum_{j=1}^{p} \log(1 + \exp(-x_{ij} \cdot \sum_{k=1}^{r} W_{ik}C_{jk})) \tag{6}$$

Why use logistic PCA? When we have binary data, modeling the output as a linear combination of factors/components doesn't make a whole lot of sense: the data are either $x_{ij} = \{+1, -1\}$, but $\mathbf{w}_i^T \mathbf{c}_j$ could be much larger or smaller than these bounds. For a longer explanation, read up on when/why to use logistic regression instead of linear regression — the reasoning is exactly analogous. It is also informative to compare classic to logistic PCA on simulated binary data, as shown in the plot below:



Logistic PCA can outperform classic PCA on binary data. See Julia code here to reproduce this figure.

There are many other variations that you can come up with, each of which is tailored to different data types and characteristics:

Robust PCA. If you have outliers in your dataset, use the sum of the absolute value of the residuals (L1 loss) or a Huber loss function (Kwak, 2008). There are some alternative formulations of robust PCA, see e.g. Candes et al. (2009) and Netrapalli et al. (2014).

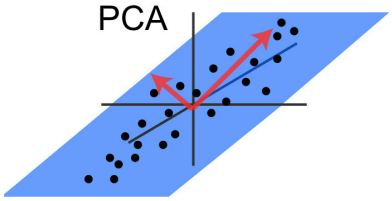
Poisson PCA and PCA on ordinal data. See Rennie & Srebro (2005) for some discussion of appropriate loss functions.

Zero-Inflated dimensionality reduction. Some datasets, such those from single-cell RNAseq, have more zero entries than would be expected under a Poisson noise model. This can arise from technical variability — mRNA is fragile, and lowly expressed genes have less starting material, leading to "dropout" of lowly expressed genes to zero. Pierson & Yau (2015) develop a model to account for this flavor of noise, and their work can be mapped onto the optimization framework described in this post.

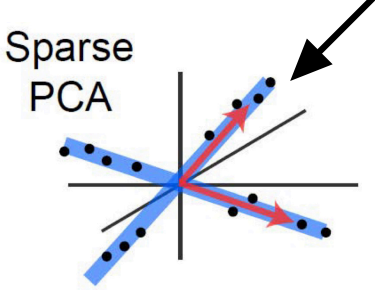
All of these methods have the same basic flavor:

- We start with a 2D array of data $\mathbf{X}$
- We define some cost function or objective function that measures the fit (e.g. least-squares or logistic loss)
- If we want, we can add terms to the cost function regularize the problem (e.g. to encourage sparsity or enforce nonnegativity)
- We optimize two smaller matrices W and C^T so that their product reconstructs the data as best as possible.
- The optimization problem is biconvex (unless the regularization terms or constraints aren't convex) suggesting alternating minimization as a reasonable optimization procedure

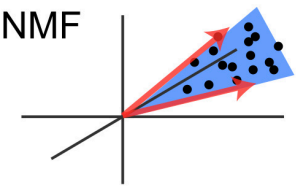
6. 矩阵分解各方法之间的关系



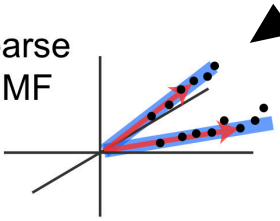
$$\underset{\mathbf{U},\mathbf{V}}{\text{minimize}} \quad \|\mathbf{X}-\mathbf{U}\mathbf{V}^T\|_F^2$$
$$\text{subject to} \quad \mathbf{U}^T\mathbf{U}=\mathbf{V}^T\mathbf{V}=\mathbf{I}$$



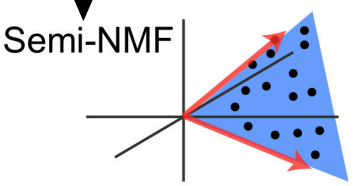
$$\underset{\mathbf{U},\mathbf{V}}{\text{minimize}} \quad \|\mathbf{X}-\mathbf{U}\mathbf{V}^T\|_F^2+\lambda_u\sum_{i=1}^{n}\|\mathbf{u}_{i:}\|_1+\lambda_v\sum_{j=1}^{m}\|\mathbf{v}_{j:}\|_2^2$$



$$\underset{\mathbf{U},\mathbf{V}}{\text{minimize}} \quad \|\mathbf{X}-\mathbf{U}\mathbf{V}^T\|_F^2$$
$$\text{subject to} \quad \mathbf{U}\geq 0,\ \mathbf{V}\geq 0$$

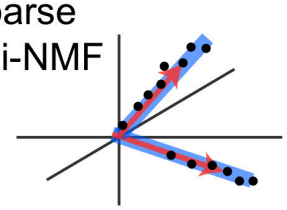


$$\underset{\mathbf{U},\mathbf{V}}{\text{minimize}} \quad \|\mathbf{X}-\mathbf{U}\mathbf{V}^T\|_F^2+\lambda_u\sum_{i}\|\mathbf{u}_{i:}\|_1$$
$$\text{subject to} \quad \mathbf{U}\geq 0,\ \mathbf{V}\geq 0$$

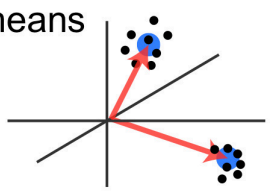


$$\underset{\mathbf{U},\mathbf{V}}{\text{minimize}} \quad \|\mathbf{X}-\mathbf{U}\mathbf{V}^T\|_F^2$$
$$\text{subject to} \quad \mathbf{U}\geq 0$$

NMF is a nice trade off between PCA & k-mean.



$$\underset{\mathbf{U},\mathbf{V}}{\text{minimize}} \quad \|\mathbf{X}-\mathbf{U}\mathbf{V}^T\|_F^2+\lambda_u\sum_{i}\|\mathbf{u}_{i:}\|_1$$
$$\text{subject to} \quad \mathbf{U}\geq 0$$



$$\underset{\mathbf{U},\mathbf{V}}{\text{minimize}} \quad \|\mathbf{X}-\mathbf{U}\mathbf{V}^T\|_F^2$$
$$\text{subject to} \quad \mathbf{u}_{i:}\in\{\mathbf{e}_k\},\forall i$$

Combinatorial menu of models

loss functions

quadratic
(real data)

absolute
(robust to outliers)

logistic
(binary data)

Poisson
(integer data)

circular
(angular data)

regularizers/constraints

L2 norm
(small factors)

L1 norm (sparsity)
(sparse factors)

Nonnegative
(additive factors)

Derivative penalties
(smooth factors)

Further Reading

Udell et al. (2016). **"Generalized Low Rank Models."** *Foundations and Trends in Machine Learning.*

Presents one of the most general matrix factorization frameworks that includes PCA, NMF, Sparse PCA, K-means, and many others as special cases.

Essid & Ozerov (2014). **Tutorial on NMF.** *ICME 2014.*

http://perso.telecom-paristech.fr/~essid/teach/NMF_tutorial_ICME-2014.pdf

A comprehensive overview of applications and extensions of NMF

Gillis (2011). **Nonnegative Matrix Factorization: Complexity, Algorithms, and Applications.** *PhD thesis, Université Catholique de Louvain.*

A very comprehensive thesis placing greater focus on the algorithmic aspects of NMF. Also see more recent work from Gillis.

Some things you maybe didn't know about PCA

PCA overfits to noise if $p > n$ (i.e. it is an inconsistent estimator of the subspace of maximal variance).

When solving linear systems of equations the number of equations must be greater than then number of unknown variables. In linear regression, this means that we need more observations than unknown variables ($n > p$). We've seen that PCA is closely related to regression, and so it should come as no big surprise that PCA runs into problems when $p > n$. Intuitively, each dimension/feature has some noise associated with it, and we need more observations than parameters to reliably tease apart the signal from the noise.

One way to potentially get around this problem is to use sparse PCA (Johnston & Lu, 2009), although this assumes that your dataset is well-represented in a sparse basis. Moreover, you shouldn't blindly assume that L1 regularization will produce the *correct* sparsity pattern (Su et al., 2015; Advani & Ganguli, 2016).

There is a very good and simple procedure to determine how many principal components to keep

A primary motivation behind PCA is to use as few components as possible to *reduce* the dimensionality of the data we are working with. Thus, we are often interested in truncating PCA — keep only the top k components and throw away the rest. There are at least two reasons for this:

- Truncating gives us a sense of how complex the dataset is. If the top two principal components capture a large majority of variance, then the dataset is more-or-less two-dimensional.[6]
- Truncating denoises the data. The conceptual connection of PCA to regression is again helpful here — PCA is analogous to fitting a smooth curve through noisy data. Similar intuition is given by figure 2 in this blog post, in which a rank-1 approximation gives a smooth, less noisy, representation of the data.

The question then becomes, how do we choose where to truncate? This used to be one of those classic questions with an unsatisfying answer… Basically, eyeball it.

Gavish & Donoho (2014) present a long overdue result on this problem and their answer is surprisingly simple and concrete. Essentially, the optimal[7] procedure boils down to estimating the noise in the dataset, σ, and then throwing away all components whose singular values are below a specified threshold. For a square $n \times n$ matrix, this threshold is:

$$\lambda = \frac{4\sigma\sqrt{n}}{\sqrt{3}}$$

There is a similar threshold for non-square datasets explained in the paper. As with any theoretical study, the result comes with a few assumptions and caveats,[8] but their work appears robust and useful in practice.

Edit: Thanks to Jonathan Pillow for pointing out a Bayesian alternative outlined here: *Minka (2000). Automatic choice of dimensionality for PCA*

PCA becomes non-trivial to solve when data entries are missing

After thinking about these topics for a while, I found it pretty incredible that PCA works at all. In the first place, it is pretty special any time you can provably and analytically solve a nonconvex optimization problem.

The specialness of PCA breaks down even under pretty mild perturbations. Ilin & Raiko (2010) discuss a nice illustration of this point. Consider the case where some subset of data entries are not observed $x_{ij} = \mathrm{NA}$. Even if you keep the ordinary PCA objective function, a number of problems arise:

- There is no analytical solution because the data covariance matrix is nontrivial to estimate
- The objective function contains local minima (unlike in classic PCA, where there are only saddle points and one global minimum). Thus, it is difficult to certify that the output of your optimization problem is true solution to the problem.
- There is no analytical solution even for the bias term, in contrast to classic PCA where the bias equals column-wise mean of the data matrix.

This last one is particularly jarring. It feels so natural to mean-center the data that is easy to forget that this is not always justified. The task of estimating missing data entries is known as *matrix completion* and is an important problem in the machine learning community (see *Netflix Prize*, Candes & Recht, 2008). Alternating minimization is a common approach for solving these problems (e.g., Jain et al., 2013).

8. 对 loss func 的优化方法

We will start with **quadratically regularized PCA**, which is similar to ridge regression. The basic idea is to penalize the squared Euclidean length (L2 norm) of the rows of W and C so that they don't get too large:

$$\underset{W,C}{\text{minimize}} \qquad \|\mathbf{X} - WC^T\|_F^2 \; + \; \gamma \sum_{i=1}^{n} \|\mathbf{w}_i\|_2^2 \; + \; \gamma \sum_{j=1}^{p} \|\mathbf{c}_j\|_2^2 \tag{3}$$

Madeleine Udell's thesis shows that the answer to this problem is very similar to classic PCA and can be solved analytically using the singular value decomposition (SVD).

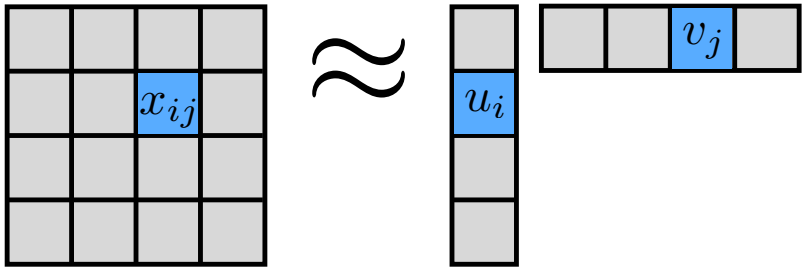
Interestingly, the rest of the PCA variants listed in this post cannot be analytically solved. In fact, PCA and quadratically-regularized PCA are quite *special cases of nonconvex optimization problems that we can solve exactly*. In practice, we can still fit the rest of these models using standard techniques like gradient descent. Even better, we can exploit the fact that ***these optimization problems are biconvex***. That is, if we treat W as a fixed constant and optimize over C then the problem is convex, and vice versa. This suggests the alternating minimization algorithm which can work very well in practice. In rough pseudocode:

Alternating minimization:

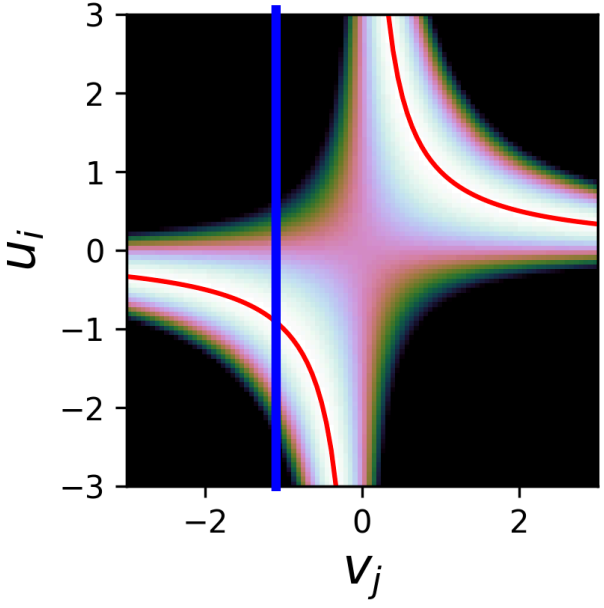
choose initial starting points $W^{(0)}$ and $C^{(0)}$
$n \leftarrow 0$
while not converged
    $W^{(n+1)} \leftarrow$ minimize over W while holding $C = C^{(n)}$ constant.
    $C^{(n+1)} \leftarrow$ minimize over C while holding $W = W^{(n+1)}$ constant.
    $n \leftarrow n + 1$
end while

Again, the idea here is that the sub-problems (4) and (5) are easy to optimize because they are convex. It isn't necessary to minimize the sub-problems to completion, in fact it can work better to take just take alternating gradient steps for each sub-problem.

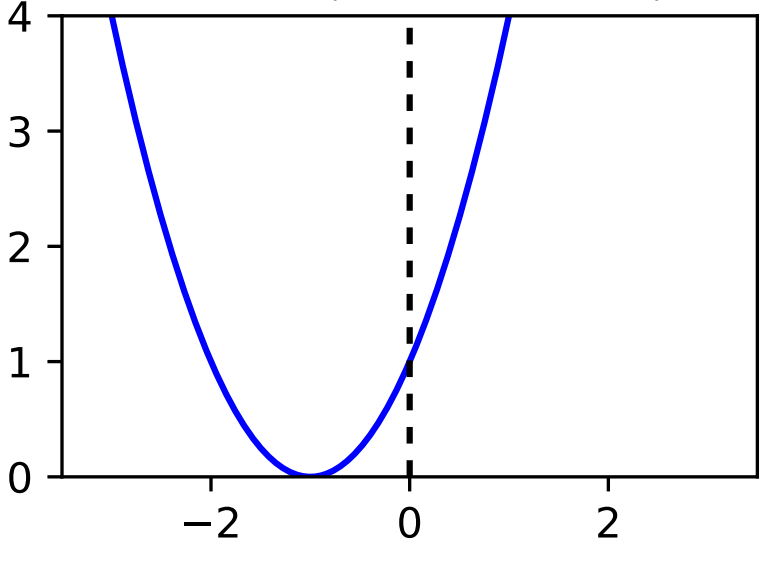
Consider the PCA loss for a single matrix element



$$\ell_{ij}(u_i, v_j) = (x_{ij} - u_i v_j)^2$$



Convex in **u** when **v** is fixed as constant (and vice versa)



Fitting PCA in 10 lines of MATLAB

```
K = 3; % number of components
data = randn(100,K) * randn(K, 101);
[M, N] = size(data);
U = randn(M, K); % initial guess for U

for iteration = 1:10
    Vt = U \ data; % Update V (fixed U)
    U = data / Vt; % Update U (fixed V)
    loss(iteration) = norm(data - U*Vt, 'fro');
end
```

Properties of PCA

PCA is one of the few examples of a nonconvex problem* that can be provably solved in polynomial time

Can prove that all local minima are solutions.

All non-optimal critical points are saddle points or maxima.

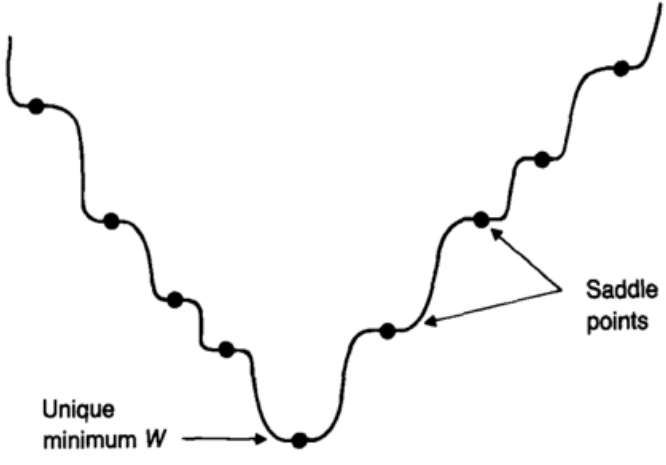


FIGURE 2. The landscape of *E*.

(Baldi & Hornik, 1989).

* with a bit of work you can formulate a convex optimization problem whose solution also solves the PCA problem:
http://www.stat.cmu.edu/~ryantibs/convexopt/lectures/nonconvex.pdf

NMF can also be solved by alternating minimization

Each step is ***nonnegative least squares*** problem

$$\mathbf{U} \leftarrow \underset{\widetilde{\mathbf{U}} \geq 0}{\operatorname{argmin}} \|\mathbf{X} - \widetilde{\mathbf{U}}\, \mathbf{V}^T\|_F^2$$

$$\mathbf{V} \leftarrow \underset{\widetilde{\mathbf{V}} \geq 0}{\operatorname{argmin}} \|\mathbf{X} - \mathbf{U}\, \widetilde{\mathbf{V}}^T\|_F^2$$

- Convex problem
- Specialized, fast optimization methods
  (e.g. Kim & Park, 2008)

In MATLAB:

```
x = lsqnonneg(A, b);
```

In Python:

```
import scipy.optimize
x = scipy.optimize.nnls(A, b)
```

Properties of PCA

Rotation problem limits interpretability. However, it also allows us to organize factors to have convenient properties.

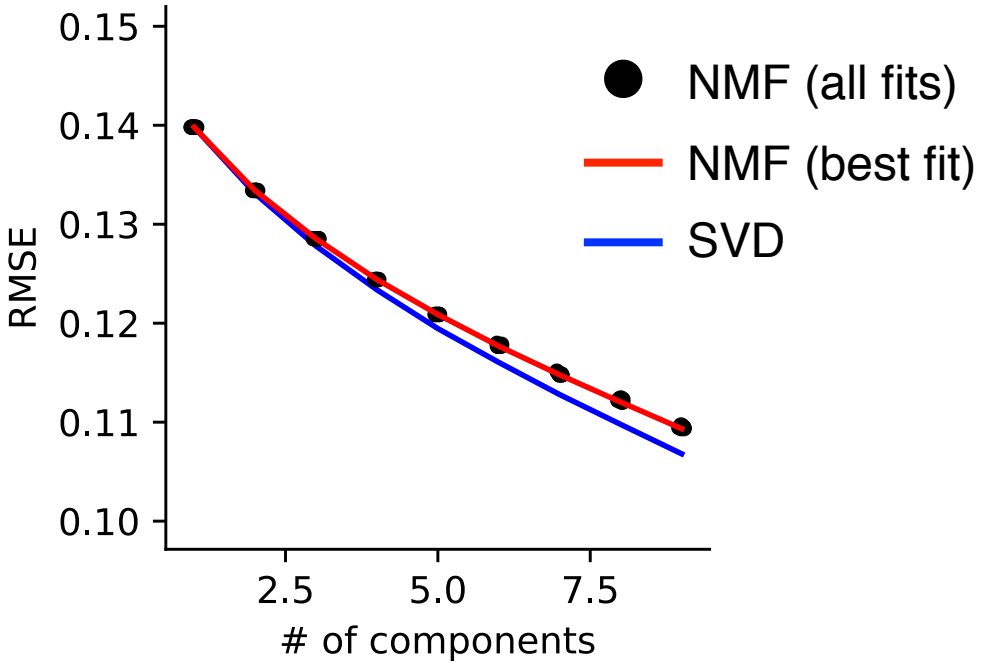
Canonically, choose factors to be orthogonal and order them by variance explained.

Eckart-Young Theorem: solution given by truncated singular value decomposition (SVD)

Consequence: the solution with R components is contained in the solution with R+1 components.

9. 评估

Scree Plot — How well am I fitting the data?



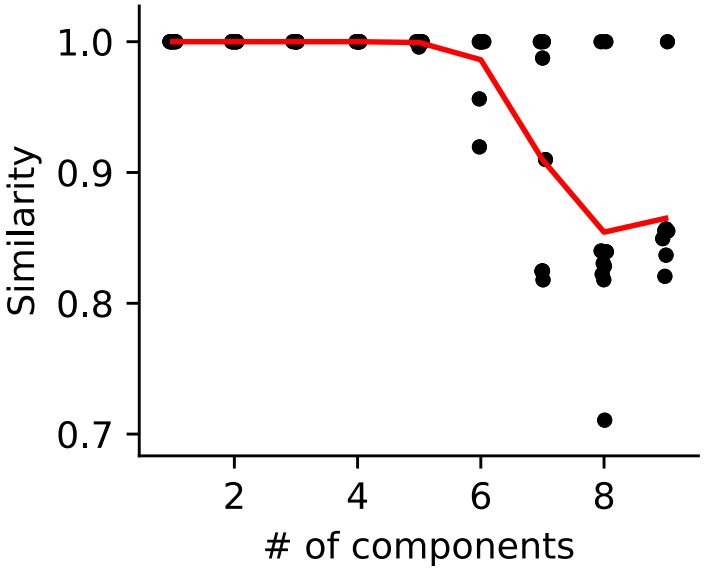
Interpretation: NMF converges to similar error from different initializations, and nearly achieves the optimal lower bound on performance set by SVD.

Similarity Plot — Are there multiple solutions that fit the data equally well?

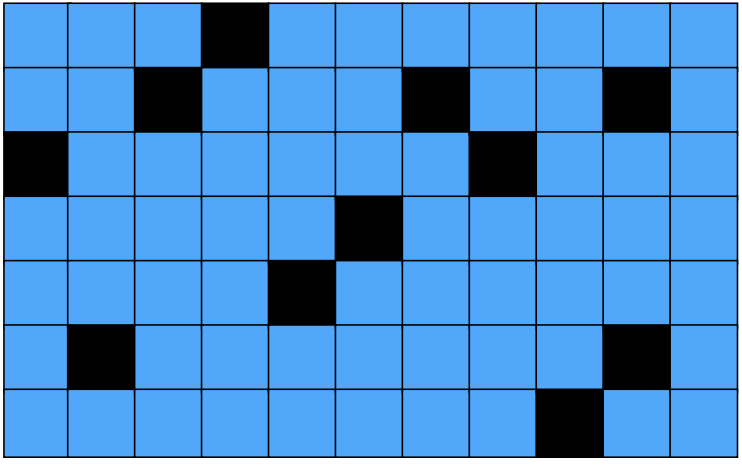
Define the similarity of two factor matrices as:

$$S(\mathbf{U}, \mathbf{U}') = \max_{\Pi} \frac{1}{r} \mathrm{Tr}\left[\mathbf{U}^T \mathbf{U}' \Pi\right]$$

where Π, is an r x r permutation matrix.



Cross-Validation



training data

held-out data

Holding out data at random for cross-validation draws a connection to the well-studied matrix completion problem (see e.g. Candès & Recht, 2009)