Organizational issues

- 1st prog. assignment's DL this Sunday
  - tomorrow's last tutorial on that one
- Next prog. assignment released later this week
  - DL 12 July
- If you want to meet with tutor outside tutorial, **send email to agree the meeting**

Chapter 3

Column and Column-Row Decompositions

max planck institut
informatik

Column Decompositions

Boutsidis et al. 2010; Strauch 2014

Motivation

- SVD is often hard to interpret and yields dense factorizations
  - NMF tries to address these problems with varying success
- But if original data is sparse & easy to interpret, why not use it in the decompositions?

The CX decomposition

- In the **CX decomposition** we are given a matrix $\boldsymbol{A}$ and a rank k, and we need to select k **columns** of $\boldsymbol{A}$ into matrix $\boldsymbol{C}$ and build matrix $\boldsymbol{X}$ s.t. we minimize $\|\boldsymbol{A} - \boldsymbol{C}\boldsymbol{X}\|_\xi$
  - ξ is either F or 2
  - A.k.a. **column subset selection problem** (CSSP)

Why CX?

- The columns of ***C*** preserve the original interpretation of columns of ***A***
    - Even complex constraints are satisfied if the original data satisfied them
- Feature selection
    - Selects the columns that can be used to explain the rest
    - Compare to the dimensionality reduction

Alternative target function

- Building $\boldsymbol{C}$ is the hard part of CX decompositions
- Given $\boldsymbol{A}$ and $\boldsymbol{C}$, $\boldsymbol{X}$ can be computed with the pseudo-inverse
  - $\boldsymbol{X} = \boldsymbol{C}^{+}\boldsymbol{A}$
  - Alternative target function for CX:
    minimize $\|\boldsymbol{A} - \boldsymbol{C}\boldsymbol{C}^{+}\boldsymbol{A}\|_{\xi} = \|\boldsymbol{A} - \boldsymbol{P}_{\boldsymbol{C}}\boldsymbol{A}\|_{\xi}$

How to select C?

- Exhaustive: try all $\binom{m}{k}$ subsets of columns
  - Not very scalable
- Try to select the columns in a clever way
  - But how?
- Sample columns w.r.t. carefully selected probabilities
  - Avoids deterministic worst-case scenarios

Related idea: RRQR

- The **rank-revealing QR** (RRQR) factorization of matrix $\boldsymbol{A}$ is

k-by-k upper-triangular w/ positive diagonal

k-by-$(m–k)$

n-by-n orthogonal

$(n–k)$-by-$(m–k)$

$$\boldsymbol{A\Pi} = \boldsymbol{QR} = \boldsymbol{Q}\begin{pmatrix} \boldsymbol{R}_{11} & \boldsymbol{R}_{12} \\ 0 & \boldsymbol{R}_{22} \end{pmatrix}$$

Permutation matrix

that satisfies

n-by-m

kth singular value of $\boldsymbol{A}$

$$\frac{\sigma_k(\boldsymbol{A})}{p_1(k,m)} \leq \sigma_{\min}(\boldsymbol{R}_{11}) \leq \sigma_k(\boldsymbol{A})$$

$$\sigma_{k+1}(\boldsymbol{A}) \leq \sigma_{\max}(\boldsymbol{R}_{22}) \leq p_2(k,m)\sigma_{k+1}(\boldsymbol{A})$$

Some polynomial on k and m

CX and RRQR

- Let $\boldsymbol{A\Pi} = \boldsymbol{QR}$ and let $\boldsymbol{\Pi}_k$ be the first k columns of $\boldsymbol{\Pi}$ and $\boldsymbol{C} = \boldsymbol{A\Pi}_k$ some k columns of $\boldsymbol{A}$
  - Now $||\boldsymbol{A} - \boldsymbol{P_C A}||_\xi = ||\boldsymbol{R}_{22}||_\xi$, $\xi = F$ or 2
  - In particular $||\boldsymbol{A} - \boldsymbol{P_C A}||_2 \leq p_2(k, m)||\boldsymbol{A} - \boldsymbol{A}_k||_2$
    - $\boldsymbol{A}_k = \boldsymbol{U}_k \boldsymbol{\Sigma}_k \boldsymbol{V}_k^T$ (truncated SVD)
    - $\boldsymbol{CX}$ is $p_2(k, m)$-approximation to SVD

Computing CX by sampling

- Let $\boldsymbol{A} = \boldsymbol{U\Sigma V}^T$ be the input and its SVD and $\boldsymbol{V}_k$ the truncated $\boldsymbol{V}$
- Sample columns of $\boldsymbol{A}$ with replacement
  - Probability p_j for selecting column j is
    $$p_j = \|(\boldsymbol{V}_k^T)_j\|_2^2 / k$$
  - Sample $O(k^2 \log(1/\delta)/\epsilon^2)$ columns and repeat log(1/δ) times returning the least-error sample

Notes on sampling

- We can prove that

$$||\boldsymbol{A} - \boldsymbol{P_C A}||_F \leq (1 + \epsilon)||\boldsymbol{A} - \boldsymbol{A}_k||_F$$

with probability at least $1 - \delta$

  - Notice that $\boldsymbol{C}$ has much more than k columns
    - $O(k^2 \log(1/\delta)/\epsilon^2)$ with large hidden constants

Why does sampling work?

- Intuitively, if $\boldsymbol{A}$ is of low rank ($k \ll n$), $\boldsymbol{A}$ should have many almost-similar columns
- If we sample many columns enough, we should get a representative for each set of similar columns

  ⇒ We need to sample more columns than the rank

  - Or our error depends on the rank…

CX with exact k

- Construct larger-than-k CX decomposition as above for $c = O(k \log k)$ columns
  - Let $\boldsymbol{\Pi}_1$ be the m-by-c matrix that selects c columns s.t. $\boldsymbol{C} = \boldsymbol{A}\boldsymbol{\Pi}_1$
  - Let $\boldsymbol{D}_1$ be c-by-c diagonal s.t. if jth column is selected on round i, $(\boldsymbol{D}_1)_{ii} = (cp_j)^{-1/2}$

From $\boldsymbol{A} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^T$

- Run RRQR algorithm for $\boldsymbol{V}_k^T\boldsymbol{\Pi}_1\boldsymbol{D}_1$ to select exactly k columns of $\boldsymbol{V}_k^T\boldsymbol{\Pi}_1\boldsymbol{D}_1$ with matrix $\boldsymbol{\Pi}_2$ (c-by-k)
  - return $\boldsymbol{C} = \boldsymbol{A}\boldsymbol{\Pi}_1\boldsymbol{\Pi}_2$

Notes on the exact-k CX

- $\Pr[||\boldsymbol{A} - \boldsymbol{P_C A}||_F \leq \Theta(k \log^{1/2} k)||\boldsymbol{A} - \boldsymbol{A}_k||_F] \geq 0.8$
- The sampling phase still requires really many columns (high hidden constants)
  - But in practice something like $c = 5k$ works
- Any RRQR algorithm can be used for the second step
  - But the analysis depends on the chosen algorithm

Non-Negative CX

Strauch 2014

Motivation

- If data is non-negative, so is $\boldsymbol{C}$
  - But $\boldsymbol{X}$ can contain negative values in standard CX
  - Non-negative $\boldsymbol{X}$ yields “parts-of-whole” interpretation similar to NMF
    - Selected columns are “pure” while others are mixtures of the pure columns
  - Non-negativity also improves sparsity

The non-negative CX decomposition

- In the **non-negative CX decomposition** (NNCX) we are given a non-negative matrix $\boldsymbol{A}$ and a rank k, and we need to select k **columns** of $\boldsymbol{A}$ into matrix $\boldsymbol{C}$ and build a non-negative matrix $\boldsymbol{X}$ s.t. we minimize $||\boldsymbol{A} - \boldsymbol{C}\boldsymbol{X}||_F$

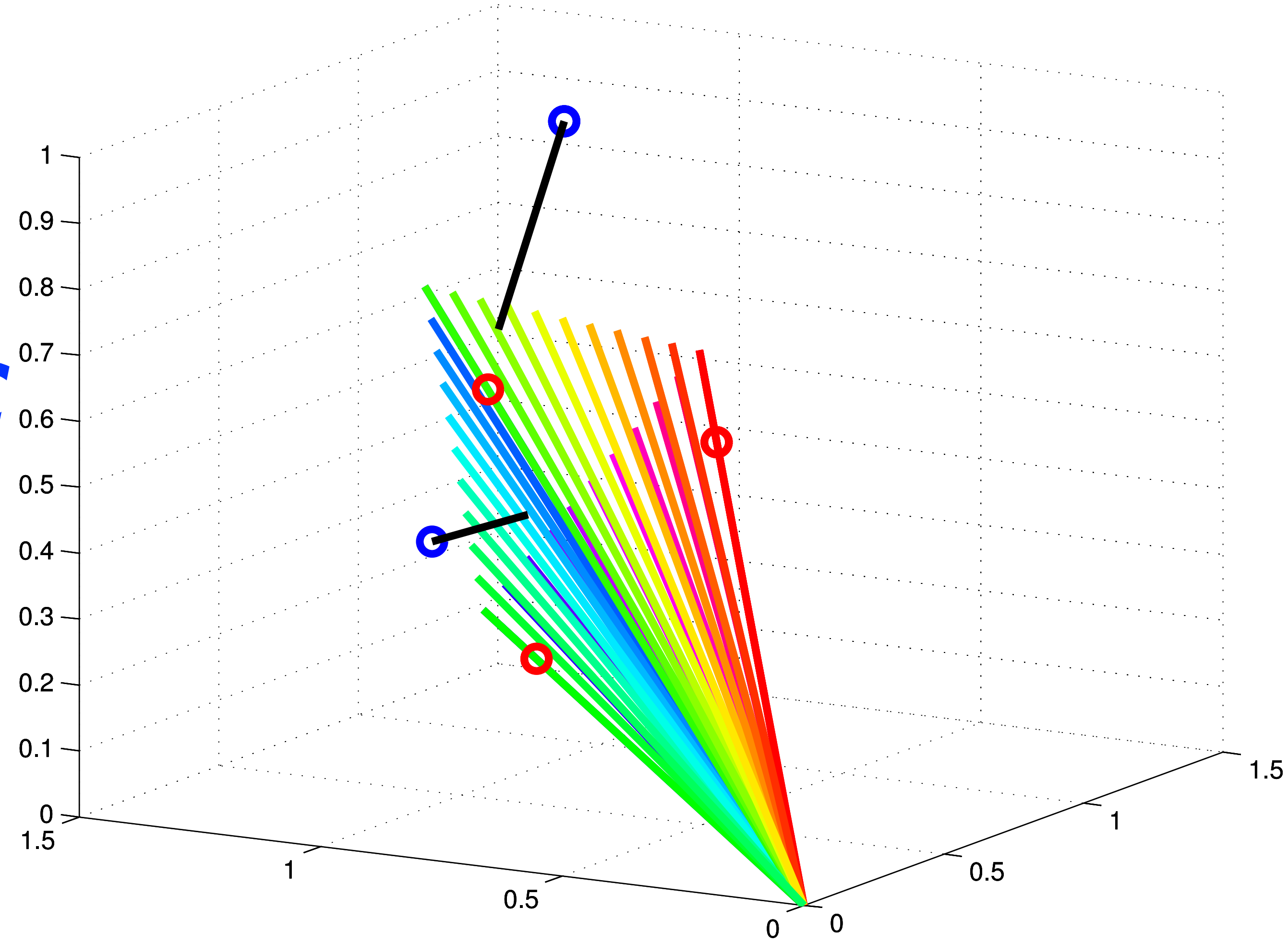
Geometry of NNCX

Columns in ***C***

Columns not in ***C***

Convex cone

Projections



Cones and columns

- Consider the cone spanned by columns of $\boldsymbol{A}$, cone($\boldsymbol{A}$)
  - If removing column j of $\boldsymbol{A}$ changes the cone, that column is **extremal**
    - Otherwise it is **internal**
  - Selecting all extremal columns to $\boldsymbol{C}$ gives us $\boldsymbol{A} = \boldsymbol{CX}$ with nonnegative $\boldsymbol{X}$

Algorithm for NNCX

- When we cannot select all extremal columns, we must choose which of them to select
  - Our goal is to maximize the volume of the convex cone
  - Finding the extremal columns is not easy
  - Given the columns, we must compute the non-negative projection

The convex_cone algorithm

- Set $\boldsymbol{R} \leftarrow \boldsymbol{A}$
- **repeat**
  - Select column $\boldsymbol{c}$ with highest norm in the residual $\boldsymbol{R}$
    - Normalize $\boldsymbol{c}$ to unit norm
  - Solve nonnegative $\boldsymbol{x}$ that minimizes $||\boldsymbol{R} - \boldsymbol{c}\boldsymbol{x}^T||$
  - Set $\boldsymbol{R} \leftarrow \boldsymbol{R} - \boldsymbol{c}\boldsymbol{x}^T$
- **until** k columns are selected
- Set $\boldsymbol{C}$ to the columns of $\boldsymbol{A}$ corresponding to the selected $\boldsymbol{c}$ and solve nonegative $\boldsymbol{X}$ minimizing $||\boldsymbol{A} - \boldsymbol{C}\boldsymbol{X}||_F$

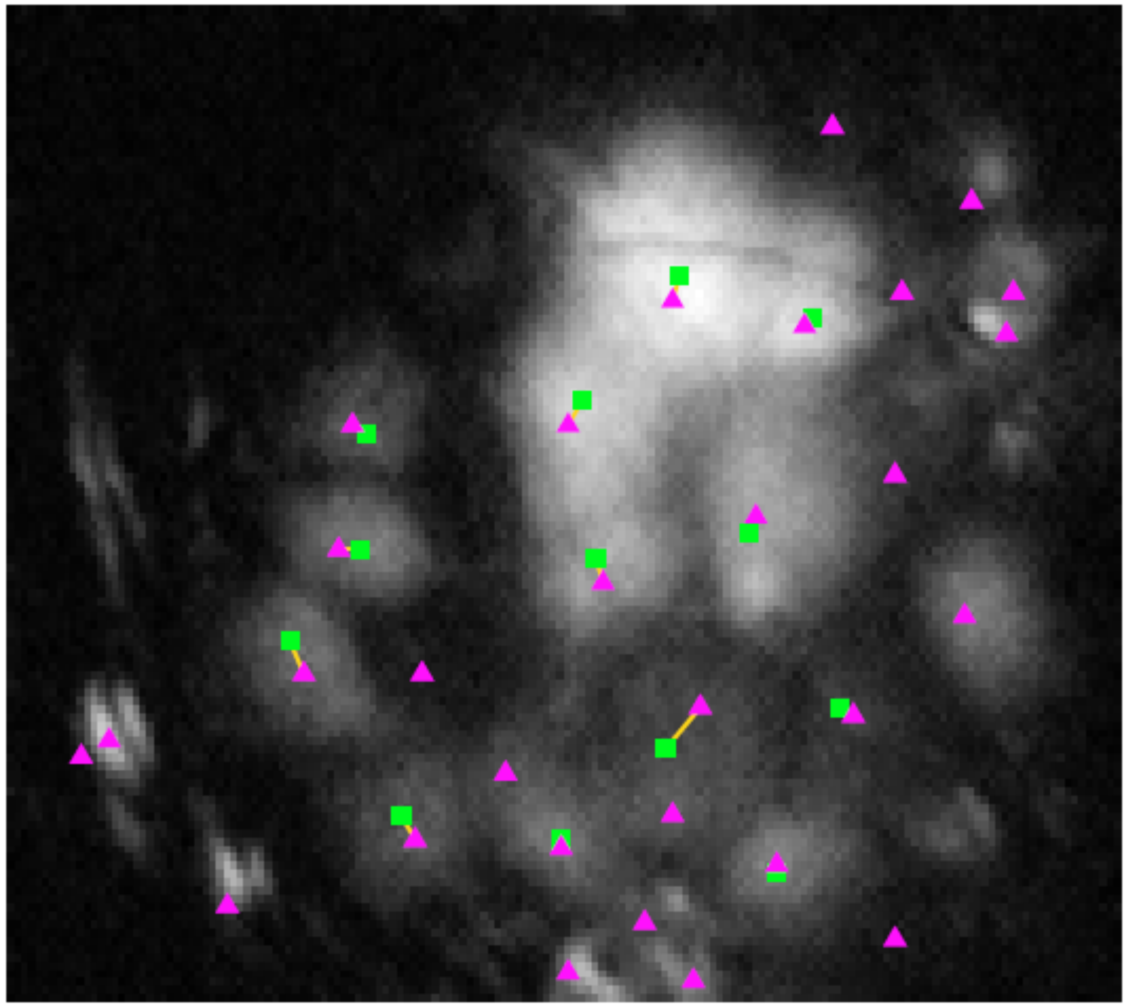
Solving for non-negative *X*

- Given ***C***, finding non-negative ***X*** is the same as with NMF
  - Convex optimization with linear constraints
  - Or truncated-to-zero pseudo-inverse

Application: Neuroimaging

- Record brain cell activity over time
  - Every row is one frame
- Assume some columns contain the pure glomerulus signal
  - ***C*** identifies these signals
  - ***X*** explains how the signals are mixed in the brain images

Movie frame with real and found locations marked

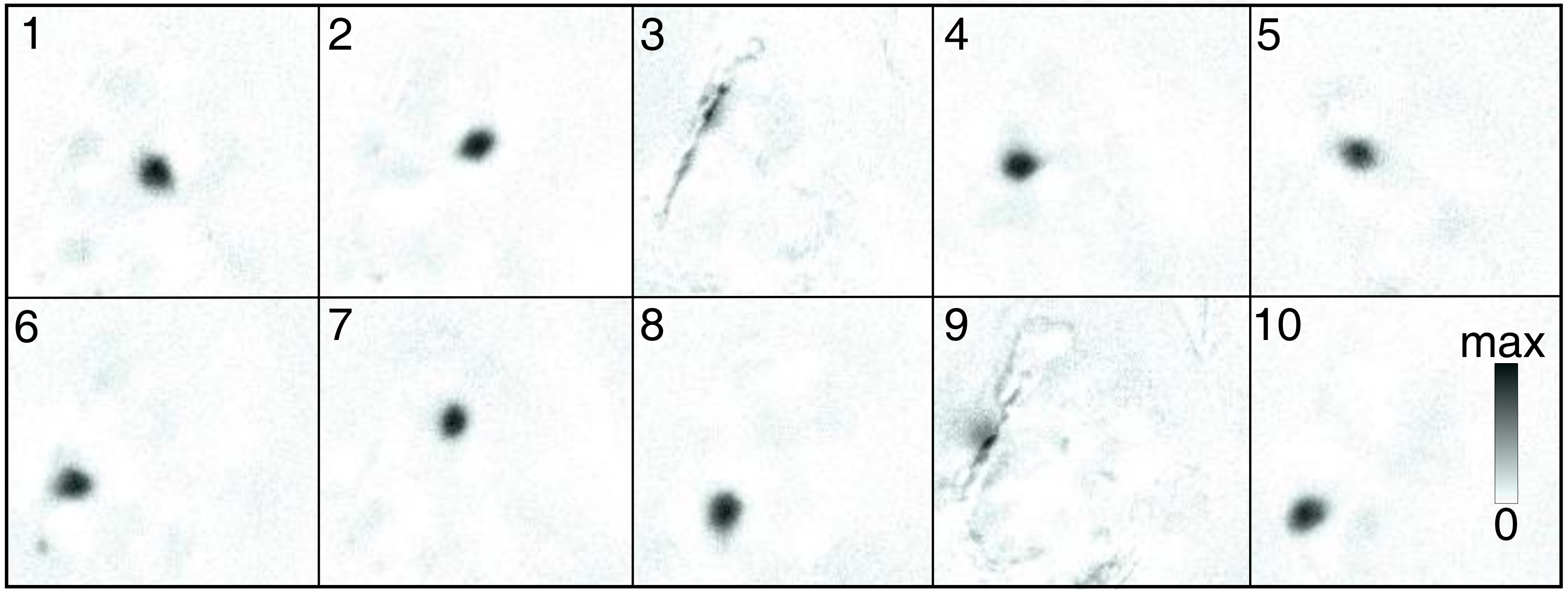


Human expert

Algorithm

Strauch 2014

Application cont’d



Top-10 rows of X from NNCX decomposition shows the shape and location of glomeruli

Column-Row Decompositions

Leskovec et al. chapter 11.4; Skillicorn chapter 3.6.2

The CUR decomposition

- In the **CUR decomposition** we are given matrix $\boldsymbol{A}$ and integers c and r, and our task is to select c columns of $\boldsymbol{A}$ to matrix $\boldsymbol{C}$ and r rows to matrix $\boldsymbol{R}$, and build c-by-r matrix $\boldsymbol{U}$ minimizing $\|\boldsymbol{A} - \boldsymbol{CUR}\|_F$
  - Often $c = r = k$

Why CUR?

- If selecting the actual columns in CX is good, selecting the actual columns and rows must be even better
  - We find prototypical columns *and rows*
  - ***U*** is usually small, so if ***C*** and ***R*** are sparse, storing CUR takes little space

Solving CUR: general idea

- CUR is two-sided CX
- Simple algorithm idea:
  - Solve CX for $\boldsymbol{A}$ and $\boldsymbol{A}^T$ and solve for $\boldsymbol{U}$ given $\boldsymbol{C}$ and $\boldsymbol{R}$
    - $\boldsymbol{U} = \boldsymbol{C}^+\boldsymbol{A}\boldsymbol{R}^+$
- Better algorithms take into account the columns selected to $\boldsymbol{C}$ when computing $\boldsymbol{R}$

Simple CUR algorithm

- Sample columns proportional to their L_2-norm
- Sample rows proportional to their L_2-norm
- Build $\boldsymbol{W} = \boldsymbol{A}[R,C]$ (the sub-matrix of columns in $\boldsymbol{R}$ and rows in $\boldsymbol{C}$)
  - Let $\boldsymbol{W} = \boldsymbol{X\Sigma Y}^T$ be an SVD of $\boldsymbol{W}$, and set $\boldsymbol{U} \leftarrow \boldsymbol{Y}(\boldsymbol{\Sigma}^+)^2\boldsymbol{X}^T$

Fancier CUR algorithm

- Find $\boldsymbol{C}$ similar to exact-k CX earlier
  - Sample $O(k/\epsilon)$ additional columns
- Find $\boldsymbol{Z} \in \text{span}(\boldsymbol{C})$, $\boldsymbol{Z}^T\boldsymbol{Z} = \boldsymbol{I}$, such that $||\boldsymbol{A}^T - \boldsymbol{A}^T\boldsymbol{Z}\boldsymbol{Z}^T||_F \leq (1 + O(\epsilon))||\boldsymbol{A} - \boldsymbol{C}\boldsymbol{X}^*||_F$
  - Use $\boldsymbol{Z}$ to get the probabilities for sampling O(k log k) rows of $\boldsymbol{A}$ and reduce that to O(k) rows
  - Sample $O(k/\epsilon)$ additional rows
- Set $\boldsymbol{U} = \boldsymbol{X}^*\boldsymbol{Z}^T\boldsymbol{A}\boldsymbol{R}^+$

Boutsidis & Woodruff 2014

Comments on the Boutsidis & Woodruff algorithm

- Slight variations of the above algorithm achieve:
  - selects the smallest number of rows and columns for $(1+\epsilon)$ approximation
  - matrix $\boldsymbol{U}$ has the smallest possible rank

CX and CUR summary

- Rows and columns of the original data should be interpretable
  - Also admit local constraints in the data
- CX and CUR decompositions are forms of feature selection
  - Applications when we need “prototypical” rows and columns

Literature

- Drineas, Mahoney & Muthukrishnan (2006): *Subspace sampling and relative-error matrix approximation: Column-based methods*. In APPROX/RANDOM ’06
- Boutsidis, Mahoney & Drineas (2010): *An improved approximation algorithm for the column subset selection problem*. arXiv:0812.4293v2
- Boutsidis & Woodruff (2014): *Optimal CUR matrix decompositions*. arXiv:1405.7910v2
- Strauch (2014): *Column subset selection with applications to neuroimaging data.* PhD thesis, U. Konstanz