Low Rank approximation
Lecture 2

F. M. Faulstich

01/12/2024

The SVD

- Recall the SVD of $\mathbf{A} \in \mathbb{F}^{m \times n}$ is given by

$$\mathbf{A} = \mathbf{U}\boldsymbol{\Sigma}\mathbf{V}^* = \sum_{i=1}^{\min(m,n)} \sigma_i \mathbf{u}_i \mathbf{v}_i^*$$

  where $\mathbf{U} \in \mathbb{F}^{m \times m}$, $\mathbf{V} \in \mathbb{F}^{n \times n}$ are orthonormal, and $\boldsymbol{\Sigma} \in \mathbb{F}^{m \times n}$ is diagonal.
  Columns of $\mathbf{U}$ are the left singular vectors
  Columns of $\mathbf{V}$ are the right singular vectors
- The rank of $\mathbf{A} \in \mathbb{F}^{m \times n}$ is given by the number of non-zero singular values

The SVD

- One construction: diagonalize

$$\mathbf{M} = \begin{pmatrix} \mathbf{0} & \mathbf{A} \\ \mathbf{A}^* & \mathbf{0} \end{pmatrix}$$

  The eigenvalues come in pairs $(\sigma_i, -\sigma_i)$.
  If $\mathbf{A} \in \mathbb{F}^{m \times n}$, the first m entries in the eigenvectors are the left singular vectors, and the last n entries are the right singular vectors.
- We can then compute the rank k matrix

$$\mathbf{A}_k = \sum_{i=1}^{k} \sigma_i \mathbf{u}_i \mathbf{v}_i^*$$

  for $k \leq \min(m, n)$.

Eckart-Young-Mirsky

- The matrix $\mathbf{A}_k$ is a rank k approximation to $\mathbf{A}$.
  How good of an approximation?
- EYM for Frobenius norm:
  Let $\mathbf{A} \in \mathbb{F}^{m \times n}$ with $\operatorname{rank}(\mathbf{A}) = r$. For any $1 \leq k \leq r$ we have

$$\|\mathbf{A} - \mathbf{A}_k\|_F = \inf_{\substack{\mathbf{B} \in \mathbb{C}^{m \times n} \\ \operatorname{rank}(\mathbf{B}) \leq k}} \|\mathbf{A} - \mathbf{B}\|_F = \sqrt{\sigma_{k+1}^2 + \ldots + \sigma_r^2}$$

- EYM for spectral norm:
  Let $\mathbf{A} \in \mathbb{F}^{m \times n}$ with $\operatorname{rank}(\mathbf{A}) = r$. For any $1 \leq k \leq r$ we have

$$\|\mathbf{A} - \mathbf{A}_k\| = \inf_{\substack{\mathbf{B} \in \mathbb{C}^{m \times n} \\ \operatorname{rank}(\mathbf{B}) \leq k}} \|\mathbf{A} - \mathbf{B}\| = \sigma_{k+1}$$

The SVD – The good and the bad

- Galois – eigendecompositions of generic matrices can only be done iteratively
  (equivalent to polynomial factorization)
- Practically speaking, computing the SVD takes $\mathcal{O}(mnp)$ operations where $p = \min(m, n)$
- SVD algorithms are difficult to parallelize and tend to be slower than forming other decompositions

The QR decomposition – Vanilla version

- The QR decomposition of $\mathbf{A} \in \mathbb{F}^{m\times n}$ is given by

$$\mathbf{A} = \mathbf{QR}$$

  where $\mathbf{Q} \in \mathbb{F}^{m\times m}$ is orthonormal, and $\mathbf{R} \in \mathbb{F}^{m\times n}$ is upper triangular.
- Use classical Gram-Schmidt: Store the inner products and normalization in $\mathbf{R}$, and store the orthonormal vectors in $\mathbf{Q}$.
- CGS can be unstable: Suppose that the first few columns are all essentially parallel with small errors. CGS will try to construct orthogonal vectors out of a set of essentially identical vectors! Sadly, there are a number of applications in which this is the setup ...

The QR decomposition – Column-pivoted QR

- Instead of CGS, let's try to write

$$\mathbf{AP} = \mathbf{QR}$$

where:
$\mathbf{A} \in \mathbb{F}^{m \times n}$
$\mathbf{P} \in \mathbb{F}^{n \times n}$ is a permutation matrix
$\mathbf{Q} \in \mathbb{F}^{m \times n}$ orthonormal
$\mathbf{R} \in \mathbb{F}^{n \times n}$ upper triangular

The QR decomposition – CPQR algorithm

- Initialize $\mathbf{Q}_0 = []$, $\mathbf{R}_0 = []$, $\mathbf{E}_0 = \mathbf{A}$, $p = \min(m, n)$
- for $k = 1 : p$

$$
\begin{aligned}
& j_k = \text{argmax}\{\|\mathbf{E}_{k-1}(:, \ell)\| \mid \ell = 1, ..., n\} \\
& \mathbf{q} = \mathbf{E}_{k-1}(:, j_k)/\|\mathbf{E}_{k-1}(:, j_k)\| \\
& \mathbf{r} = \mathbf{q}^*\mathbf{E}_{k-1} \\
& \mathbf{Q}_k = (\mathbf{Q}_{k-1}, \mathbf{q}) \\
& \mathbf{R}_k = \begin{pmatrix} \mathbf{R}_{k-1} \\ \mathbf{r} \end{pmatrix} \\
& \mathbf{E}_k = \mathbf{E}_{k-1} - \mathbf{q}\mathbf{r}
\end{aligned}
$$

  end for
- $\mathbf{Q} = \mathbf{Q}_p$, $\mathbf{R} = \mathbf{R}_p$, $\mathbf{P} = (j_1, ..., j_p)$

Low rank via QR

- After k steps of the previous algorithm, we have

$$\mathbf{A} = \mathbf{Q}_k \mathbf{R}_k + \mathbf{E}_k$$

  where $\mathbf{A}, \mathbf{E}_k \in \mathbb{F}^{m \times n}$, $\mathbf{Q}_k \in \mathbb{F}^{m \times k}$ and $\mathbf{R}_k \in \mathbb{F}^{k \times n}$
- The first term is of rank k, and the second term is the reminder.
- A reasonable stopping criterion would be

$$\|\mathbf{E}_k\|_F \leq \varepsilon$$

- We can use the partial CPQR to obtain a partial SVD
  1) Compute an SVD of $\mathbf{R}_k = \hat{\mathbf{U}} \boldsymbol{\Sigma} \mathbf{V}^*$ (cheap since $\mathbf{R}_k$ has k rows)
  2) Set $\mathbf{U} = \mathbf{Q}_k \hat{\mathbf{U}}$
  $\Rightarrow \mathbf{A} = \mathbf{U} \boldsymbol{\Sigma} \mathbf{V}^* + \mathbf{E}_k$

The interpolative decomposition

a.k.a. skeletonization

- The ID of $\mathbf{A} \in \mathbb{F}^{m \times n}$ is given by

$$\mathbf{A} = \mathbf{C}\mathbf{Z}$$

  where $\mathbf{C} \in \mathbb{F}^{m \times k}$ consists of k columns of $\mathbf{A}$ and $\mathbf{Z} \in \mathbb{F}^{k \times n}$ is a "well-conditioned" matrix.
- Clearly, if $\mathbf{A}$ is sparse or non-negative then $\mathbf{C}$ will also be sparse or non-negative.
  (This is not true with the QR or SVD)
- The ID typically requires less memory than QR or SVD
- The indices of the columns tell us something about the data!
  (Also physics preserving)

The interpolative decomposition

a.k.a. skeletonization

- There is also a row-based ID of $\mathbf{A} \in \mathbb{F}^{m \times n}$:

$$\mathbf{A} = \mathbf{X}\mathbf{R}$$

  where $\mathbf{X} \in \mathbb{F}^{m \times k}$ is well-conditioned and the rows of $\mathbf{R} \in \mathbb{F}^{k \times n}$ are a subset of the rows of $\mathbf{A}$.
- Finally, we do both:

$$\mathbf{A} = \mathbf{X}\mathbf{A}_s\mathbf{Z}$$

  where $\mathbf{X} \in \mathbb{F}^{m \times k}$ and $\mathbf{Z} \in \mathbb{F}^{k \times n}$ are well-conditioned, and $\mathbf{A}_s \in \mathbb{F}^{k \times k}$ is a submatrix of $\mathbf{A}$.
- The latter can be formed by taking a row-ID of the column-ID or vice versa.
- The choices of subsets of rows and columns are often referred to as skeletons

How do we compute it?

- From CPQR we obtain a set of columns that effectively span the column space. They are also, in a certain sense, pretty orthogonal to each other.
- Write: $\mathbf{AP} = \mathbf{QS}$ and set

$$\mathbf{Q} = (\mathbf{Q}_1, \mathbf{Q}_2) \quad \text{and} \quad \mathbf{S} = \begin{pmatrix} \mathbf{S}_{11} & \mathbf{S}_{12} \\ \mathbf{0} & \mathbf{S}_{22} \end{pmatrix}$$

where $\mathbf{Q}_1 \in \mathbb{F}^{m\times k}$, $\mathbf{Q}_2 \in \mathbb{F}^{m\times(n-k)}$, $\mathbf{S}_{11} \in \mathbb{F}^{k\times k}$, etc.

How do we compute it?

- Then

$$
\begin{aligned}
\mathbf{AP} &= (\mathbf{Q}_1\mathbf{S}_{11}, \mathbf{Q}_1\mathbf{S}_{12} + \mathbf{Q}_2\mathbf{S}_{22}) \\
&= \mathbf{Q}_1(\mathbf{S}_{11}, \mathbf{S}_{12}) + \mathbf{Q}_2(\mathbf{0}, \mathbf{S}_{22}) \\
&= \mathbf{Q}_1\mathbf{S}_{11}(\mathbf{1}, \mathbf{S}_{11}^{-1}\mathbf{S}_{12}) + \mathbf{Q}_2(\mathbf{0}, \mathbf{S}_{22}) \\
&= \mathbf{Q}_1\mathbf{S}_{11}(\mathbf{1}, \mathbf{T}) + \mathbf{Q}_2(\mathbf{0}, \mathbf{S}_{22}) \\
\Leftrightarrow \mathbf{A} &= \mathbf{Q}_1\mathbf{S}_{11}(\mathbf{1}, \mathbf{T})\mathbf{P}^* + \mathbf{Q}_2(\mathbf{0}, \mathbf{S}_{22})\mathbf{P}^* \\
&= \mathbf{Q}_1\mathbf{S}_{11}\mathbf{Z} + \mathbf{Q}_2(\mathbf{0}, \mathbf{S}_{22})\mathbf{P}^* \\
&= \mathbf{CZ} + \mathbf{Q}_2(\mathbf{0}, \mathbf{S}_{22})\mathbf{P}^*
\end{aligned}
$$

- What bout $\mathbf{S}_{11}^{-1}$? If $\mathbf{A}$ is at least rank k then $\mathbf{S}_{11}$ is invertible. If it is not, then change k

What about speed?

Randomized low-rank approximations

- At some point it is difficult to guarantee that the deterministic columns we chose are guaranteed to be a well-conditioned basis for the entire column space. This can be especially problematic when we don't know the rank!
- Idea (exactly rank k): random sketching.
  Apply your matrix to a suitably-scaled random matrix. With probability 1 it won't 'miss' any of the columns.
- Example:
  Consider $\mathbf{G} \in \mathbb{R}^{n \times k}$ an i.i.d. Gaussian matrix.
  Set $\mathbf{Y} = \mathbf{AG}$ and $\mathbf{A}_k = \mathbf{Y}(\mathbf{Y}^\dagger \mathbf{A})$
  ... How to calculate?

Two-stage low-rank approximation

- Given $\mathbf{A} \in \mathbb{F}^{m \times n}$ of rank k.
- Stage A: Sketch it!
  Compute an approximate basis for the range of $\mathbf{A}$, i.e., $\mathbf{Q} \in \mathbb{F}^{m \times \ell}$ orthonormal with $k \leq \ell \leq n$ s.t.

$$\mathbf{A} \approx \mathbf{Q}\mathbf{Q}^* \mathbf{A}$$

- Stage B: Classical factorization
  Compute SVD of $\mathbf{B} = \mathbf{Q}^* \mathbf{A} \in \mathbb{F}^{\ell \times n}$ (this is a much smaller matrix!)

$$\mathbf{B} = \hat{\mathbf{U}} \boldsymbol{\Sigma} \mathbf{V}^*$$

  and define $\mathbf{U} = \mathbf{Q}\hat{\mathbf{U}}$
- **All** accuracy loss and computational cost are now in Stage A!

Sketching – The good and the bad

- Obviously, the best sketching vectors are singular vectors... which would defeat the point.
- If the matrix is exactly rank k and we sketch with a matrix of size $m \times k$ then (with probability 1) the column space of Y will contain the column space of A and so (disregarding condition number issues) can be used as a sketching matrix.
- If the rank($\mathbf{A}$) is not exactly k the lower singular vectors can contaminate the entries of Y producing poor results.
  The fix? Take $k + 10$...

Sketching

- Fix a small integer p (like 10 or 50).
- For a set of $k+p$ Gaussian ransom vectors $\{\mathbf{g}_j\}$
- Apply $\mathbf{A}$ to obtain $\mathbf{y}_j = \mathbf{A}\mathbf{g}_j$
- Perform Gram-Schmidt of $\mathbf{y}_j$ to obtain $\mathbf{q}_j$
- This still requires $\mathcal{O}(mnk)$ work – though can be optimized! (matmat, matvec, etc)

Randomized range finder

- We want to find $\mathbf{Q} \in \mathbb{F}^{m\times\ell}$ with smallest ℓ s.t.

$$\|\mathbf{1} - \mathbf{Q}^*\mathbf{Q}\mathbf{A}\| \leq \varepsilon$$

  for a desired ε.
- Incrementally use the previous idea:
    1. Draw a Gaussian vector $\mathbf{g}_i$ and compute $\mathbf{y}_i = \mathbf{A}\mathbf{g}_j$
    2. Construct $\tilde{\mathbf{q}} = (\mathbf{1} - \mathbf{Q}_{i-1}\mathbf{Q}_{i-1}^*)\mathbf{y}_i$
    3. Set $\mathbf{q}_i = \tilde{\mathbf{q}}/\|\tilde{\mathbf{q}}\|$
    4. Form $\mathbf{Q}_i = (\mathbf{Q}_{i-1}, \mathbf{q}_i)$

  Continue until the desired accuracy is reached.

RSVD

- Halko, Martinsson, and Tropp [Theorem 1.1]:
  Suppose that $\mathbf{A} \in \mathbb{R}^{m \times n}$. Select a target rank $k \geq 2$ and an oversampling parameter $p \geq 2$, where $k + p \leq \min m, n$. Execute the proto-algorithm with a standard Gaussian test matrix to obtain $\mathbf{Q} \in \mathbb{R}^{m \times (k+p)}$ orthonormal. Then

$$\mathbb{E}\left(\|\mathbf{A} - \mathbf{Q}\mathbf{Q}^*\mathbf{A}\|\right) \leq \left(1 + \frac{4\sqrt{k+p}}{p-1}\sqrt{\min(m,n)}\right)\sigma_{k+1}$$

  Recall EYM:

$$\inf_{\substack{\mathbf{B} \in \mathbb{C}^{m \times n} \\ \operatorname{rank}(\mathbf{B}) \leq k}} \|\mathbf{A} - \mathbf{B}\| = \sigma_{k+1}$$

- On average, the algorithm produces a basis whose error lies within a small polynomial factor of the theoretical minimum