Fast updating algorithms for latent semantic indexing

Eugene Vecharynski
Lawrence Berkeley National Laboratory

Yousef Saad
University of Minnesota

Large-scale text mining

The targeted setting:

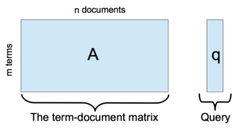
- Large-scale unstructured collection of text.
- Retrieve information (text documents) relevant to a given query.
- Text data are non-static, i.e., constantly updated.
- Consider the Latent Semantic Indexing (LSI) approach for information retrieval.

Our goal:

- Develop fast algorithms for updating the approximate Singular Value Decomposition (SVD) in the context of LSI.
- Ensure that the new updating algorithms are well-suited for large-scale data sets.
- Develop a unified framework for the SVD updating in LSI.

The vector space model

Text data and queries can be represented using linear algebra objects



- A is an m-by-n term-document matrix that encodes the data set.
- Columns of A correspond to documents, rows correspond to terms.
- An entry a_{ij} of A (term weight) essentially represents a frequency of the occurrence of the i-th term in the j-th document.
- Query q can be viewed as a separate document.

Latent Semantic Indexing (LSI)

LSI [*Deerwester et al*, 1990] is a well-established text mining technique that

- Retrieves documents based on semantic similarity rather than direct term matching.
- Requires the partial SVD of the term-document matrix

$$A \approx U_k \Sigma_k V_k^T,$$

where $\Sigma_k = \text{diag}\,\{\sigma_1, \sigma_2, \ldots, \sigma_k\} \in \mathbb{R}^{k\times k}$ contains k dominant singular values; $U_k = [u_1, \ldots, u_k] \in \mathbb{R}^{m\times k}$ and $V_k = [v_1, \ldots, v_k] \in \mathbb{R}^{n\times k}$ are matrices of the associated left and right singular vectors.

- Performs information retrieval based on the relevance scores

$$r = \text{diag}\,(\gamma_1, \ldots, \gamma_n)\left(V_k \Sigma_k^{1-\alpha}\right)\Sigma_k^{\alpha} U_k^T q,$$

where γ_j normalize the rows of $V_k\Sigma_k^{1-\alpha}$ and $\alpha \in [0, 1]$ is a splitting parameter.

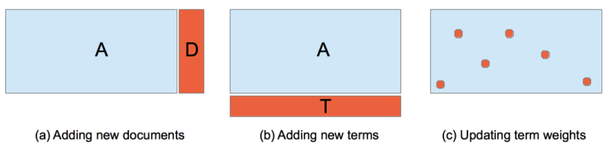
Computational challenges in LSI

Common challenges include

1. Computing partial SVD of a very large and sparse term-document matrix
   $\Rightarrow$ can be fulfilled using modern singular value solvers (e.g., SLEPc, SVDPACK)
2. Updating the approximate SVD after A is modified (e.g., by adding new documents)
   $\Rightarrow$ the Zha–Simon algorithm [SIAM J. Scient. Comput. vol. 21 (1999), pp. 782-791]

Updating problems in LSI

The common types of updates in LSI



(a) Adding new documents (b) Adding new terms (c) Updating term weights

Can the SVD factors Σ_k, U_k, and V_k be efficiently updated without recomputing the SVD?

The Zha–Simon algorithm: Adding new documents

Input: The SVD factors Σ_k, U_k, V_k and a matrix $D \in \mathbb{R}^{m\times p}$ of p new documents.
Output: Updated factors $\tilde{\Sigma}_k$, $\tilde{U}_k$, $\tilde{V}_k$.

1. Compute the QR decomposition $(I - U_k U_k^T)D = \hat{U}_D R$.
2. Construct

$$H_D = \begin{bmatrix} \Sigma_k & U_k^T D \\ 0 & R \end{bmatrix} \in \mathbb{R}^{(k+p)\times(k+p)}.$$

   Compute the SVD of H_D. Define Θ_k, F_k, and G_k to be the matrices of k dominant singular values of H_D and of the corresponding left and right singular vectors.
3. Return the updated singular triplets

$$\tilde{\Sigma}_k = \Theta_k, \;\; \tilde{U}_k = [U_k, \; \hat{U}_D] F_k, \;\; \text{and} \;\; \tilde{V}_k = \begin{bmatrix} V_k & 0 \\ 0 & I_p \end{bmatrix} G_k.$$

Restrictions of the approach for large-scale problems:

- For large updates, the cost is dominated by the SVD of H_D, which has complexity $\mathcal{O}((k+p)^3)$.
- This effect is especially pronounced for large-scale data sets, where p can be on the order of thousands or more.

We are looking for updating schemes that scale linearly in the update size, not cubically.

The Rayleigh–Ritz procedure for the singular value problem (SV-RR)

Input: $A \in \mathbb{R}^{m\times n}$, orthonormal $U \in \mathbb{R}^{m\times s_1}$ and $V \in \mathbb{R}^{n\times s_2}$. Output: $\tilde{\Sigma}_k$, $\tilde{U}_k$, $\tilde{V}_k$.

1. Construct $H = U^T A V \in \mathbb{R}^{s_1\times s_2}$.
2. Compute the SVD of H. Form matrices Θ_k, F_k, and G_k that correspond to the k dominant singular triplets of H.
3. Return $\tilde{\Sigma}_k = \Theta_k, \quad \tilde{U}_k = UF_k, \quad \tilde{V}_k = VG_k$.

- SV-RR delivers approximate left and right singular vectors $\tilde{U}_k$ and $\tilde{V}_k$ from the search subspaces $\mathcal{U} = \text{col}(U)$ and $\mathcal{V} = \text{col}(V)$.
- The SV-RR approximations are optimal:

$$\theta_i = \min_{\substack{\mathcal{S}_U \subseteq \mathcal{U}, \mathcal{S}_V \subseteq \mathcal{V}, \\ \dim(\mathcal{S}_U)+\dim(\mathcal{S}_V) = s-i+1}} \; \max_{\substack{u\in\mathcal{S}_U;\; v\in\mathcal{S}_V; \\ \|u\|=\|v\|=1}} u^T A v, \qquad i = 1, \ldots, \min(s_1, s_2).$$

LSI updating methods from the SV-RR point of view

The Zha–Simon algorithm can be viewed as the SV-RR(A_D, U, V) with

$$A_D = [A_k \; D], \;\; U = \left[U_k, \; \hat{U}_D\right], \;\; V = \begin{bmatrix} V_k & 0 \\ 0 & I_p \end{bmatrix}, \quad A_k = U_k \Sigma_k V_k^T \approx A.$$

- H_D in the Zha–Simon algorithm is the projected matrix H in the SV-RR procedure.
- Can obtain smaller H by using smaller subspaces $\Rightarrow$ compress $\hat{U}_D$ into $l << p$ vectors.

Algorithm 1: SV-RR with a compressed search subspace (singular vectors)

Input: The SVD factors Σ_k, U_k, V_k and a matrix $D \in \mathbb{R}^{m\times p}$ of p new documents.
Output: Updated factors $\tilde{\Sigma}_k$, $\tilde{U}_k$, $\tilde{V}_k$.

1. Find $l << p$ largest singular triplets of $(I - U_k U_k^T)D$, so that $(I - U_k U_k^T)D \approx X_l S_l Y_l^T$.
2. Construct

$$H_D = \begin{bmatrix} \Sigma_k & U_k^T D \\ 0 & S_l Y_l^T \end{bmatrix} \in \mathbb{R}^{(k+l)\times(k+p)}.$$

   Compute the SVD of H_D. Define Θ_k, F_k, and G_k to be the matrices of k dominant singular values of H_D and of the corresponding left and right singular vectors.
3. Return the updated singular triplets

$$\tilde{\Sigma}_k = \Theta_k, \;\; \tilde{U}_k = [U_k, \; X_l] F_k, \;\; \text{and} \;\; \tilde{V}_k = \begin{bmatrix} V_k & 0 \\ 0 & I_p \end{bmatrix} G_k.$$

- The algorithm can be viewed as the SV-RR(A_D, $\tilde{U}$, V) with

$$A_D = [A_k \; D], \;\; \tilde{U} = [U_k, \; X_l], \;\; V = \begin{bmatrix} V_k & 0 \\ 0 & I_p \end{bmatrix}, \quad A_k = U_k \Sigma_k V_k^T \approx A.$$

- The left search subspace is compressed using the SVD of $(I - U_k U_k^T)D$.
- The SVD of H_D now costs $\mathcal{O}((k+l)^2(k+p))$, which is linear in p.

Algorithm 2: SV-RR with a compressed search subspace (GKL vectors)

Input: The SVD factors Σ_k, U_k, V_k and a matrix $D \in \mathbb{R}^{m\times p}$ of p new documents.
Output: Updated factors $\tilde{\Sigma}_k$, $\tilde{U}_k$, $\tilde{V}_k$.

1. Run $l << p$ steps of the Golub–Kahan–Lanczos (GKL) procedure to generate orthogonal matrices P_l and Q_{l+1} of the left and right GKL vectors, and the bidiagonal matrix $\underline{B}_l \in \mathbb{R}^{l\times(l+1)}$, so that $(I - U_k U_k^T)D \approx P_l \underline{B}_l Q_{l+1}^T$.
2. Construct

$$H_D = \begin{bmatrix} \Sigma_k & U_k^T D \\ 0 & \underline{B}_l Q_{l+1}^T \end{bmatrix} \in \mathbb{R}^{(k+l)\times(k+p)}.$$

   Compute the SVD of H_D. Define Θ_k, F_k, and G_k to be the matrices of k dominant singular values of H_D and of the corresponding left and right singular vectors.
3. Return the updated singular triplets

$$\tilde{\Sigma}_k = \Theta_k, \;\; \tilde{U}_k = [U_k, \; P_l] F_k, \;\; \text{and} \;\; \tilde{V}_k = \begin{bmatrix} V_k & 0 \\ 0 & I_p \end{bmatrix} G_k.$$

- The algorithm can be viewed as the SV-RR(A_D, $\tilde{U}$, V) with

$$A_D = [A_k \; D], \;\; \tilde{U} = [U_k, \; P_l], \;\; V = \begin{bmatrix} V_k & 0 \\ 0 & I_p \end{bmatrix}, \quad A_k = U_k \Sigma_k V_k^T \approx A.$$

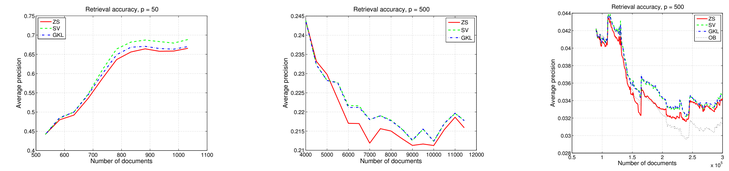
- The left search subspace is compressed using the GKL procedure for $(I - U_k U_k^T)D$.
- The SVD of H_D now costs $\mathcal{O}((k+l)^2(k+p))$, which is linear in p.

Test setting

- Compare Algorithm 1 (SV) and Algorithm 2 (GKL) to the Zha–Simon (ZS) scheme.
- Test the MEDLINE ($m = 7,014$; $n = 1,033$), NPL ($m = 7,491$; $n = 11,429$), and TREC8 ($m = 138,232$ and $n = 528,028$) collections.
- Fix t initial columns (documents) of A and add the rest in groups of p, where $t = 533$ (MEDLINE), $t = 4,000$ (NPL), and $t = 90,000$ (TREC8).
- In Algorithm 1, the l dominant singular values of $(I - U_k U_k^T)D$ are computed approximately. The matrix $(I - U_k U_k^T)D$ is never formed explicitly.

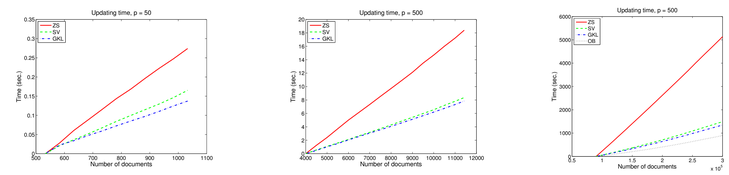
Results

Figure: Retrieval accuracy



(a) MEDLINE (b) NPL (c) TREC8

Figure: Execution time



(a) MEDLINE (b) NPL (c) TREC8

- $k = 75$ (MEDLINE), $k = 550$ (NPL), and $k = 400$ (TREC8).
- In Algorithm 1: $l = 4$ (MEDLINE), $l = 10$ (NPL), and $l = 10$ (TREC8).
- In Algorithm 2: $l = 5$ (MEDLINE), $l = 20$ (NPL), and $l = 20$ (TREC8).
- The case where $l = 0$ is reported for TREC8 (OB).

Conclusions

- Significant speedup obtained compared to the existing algorithms.
- Similar or higher retrieval accuracy.
- Future work is to assess performance on extreme-scale data sets.

Email: evecharynski@lbl.gov