High-dimensional data & scalable indexing

Outline

- Big data and high-dimensional sparse vector spaces
- Inverted indexing for scalable computation
- Review of related matrix algebra & calculus
Typical Text Mining Tasks

- Information Retrieval (search)
- Collaborative Filtering (CF)
- Link Analysis (HITS, PageRank)
- Text Categorization (TC)
- Dimensionality reduction (PCA, SVD, MF)

All use a sparse vector to represent each data point (word, document, user, item, web site, etc.), and a sparse matrix for a data collection.

A document example

I have been looking and looking for a new camera to replace our bulky, but simple and reliable (but only fair picture taker) Sony Mavica FD73. My other choice (Besides the more expensive Nikon Coolpix 3100) was the (also more expensive) Sony Cybershot P72. I recommend any of these cameras, and I was set to buy the Sony, but at the last minute I cheaped out and bought the 2100. No regrets. I bought the camera (along with 128mb memory card (the stock 16mb card will be kept in the bag as a spare) and carrying case) at the new Best Buy in Harrisburg, PA. I also bought a set of 4 Nickel-Metal Hybrid rechargeable batteries and charger at Walmart for less than $20. I keep 2 in the camera and two in the charger/in the camera bag along with the original Lithium battery pack as spares.

- This format isn't useful for many algorithms
- We can transform it into a bag of words (or features)
Typical Preprocessing of Text

- Remove stop words (500+ in English)
- Convert words to stems
  - E.g., images → image
- Canonicalize abbreviations
  - {U. S., US, U. S. A., usa, us, …} → USA
- Remove non-word symbols
  - 0-9, ",", "/", etc.
- Take the union of all terms in a corpus as the vocabulary.
**Term Weighting Schemes**

- **TF (Term Frequency - local statistic)**
  
  \[ TF(t, d) = \text{count of term } t \text{ in document } d \]

- **IDF (Inverse of Document Frequency - global statistic)**
  
  \[ IDF(t, D) = \log \left( \frac{N}{n(t, D)} + 1 \right) \]
  
  where \( D \) is a collection of \( N \) docs and \( n(t, D) \) is the count of \( t \) in \( D \).

- **TF-IDF (a popular scheme)**
  
  \[ TF-IDF(t | d, D) = TF(t, d) \times IDF(t, D) \]

---

**A Sparse-matrix View of Data (Vector Space Model)**

- Each row is a document (a bag of terms) in collection \( D \)
- Each column is a unique term (a bag of documents) in \( D \)
- Each cell is the within-document term weight (e.g., TF*IDF)
- Most of the cells of in the matrix are zeros!
### Retrieval as Matrix Multiplication

Matrix $X$ (n doc's, m words)  
Query $q$  
Similarity Scores

\[
\begin{bmatrix}
  x_{11} & x_{12} & \cdots & x_{1m} \\
  x_{21} & x_{22} & \cdots & x_{2m} \\
  \vdots & \vdots & \ddots & \vdots \\
  x_{n1} & x_{n2} & \cdots & x_{nm}
\end{bmatrix}
\begin{bmatrix}
  q_1 \\
  q_2 \\
  \vdots \\
  q_m
\end{bmatrix}
= 
\begin{bmatrix}
  y_1 \\
  y_2 \\
  \vdots \\
  y_n
\end{bmatrix}
\]

- Computing $y := Xq$ yields the similarity scores of doc’s w.r.t. the query.
- Resulting in the cosine similarities if both doc’s and query are normalized.

### Cosine Similarity

\[
cos(x_i, q) = \frac{x_{i1}q_1 + x_{i2}q_2 + \cdots + x_{im}q_m}{\|x_i\|\|q\|}
\]

- Computing $cos(x_i, q)$ takes $O(m)$ time and $O(m)$ space if using dense matrix representations
- Computing $y := Xq$ takes $O(mn)$ time and $O(mn)$ space if using a dense matrix representation of data
- Wasting most time and space on dealing with zero entries!
High-dimensional Sparse Vectors

- # of unique words in English
  - 470k entries in Webster (1993)
  - 1M+ if including all abbreviations, misspellings, etc.
- # of documents in benchmark datasets in IR
  - 870M doc's in WebClue12 (TREC), for example
- # of unique words per document on average
  - Tens or hundreds per news article, for example
- Consider a 1M-by-870M matrix: How sparse is it?
  → 99.9%+ of the entries in the vectors are zero

Outline

- Big data and high-dimensional sparse matrices
- Inverted indexing for efficient storage and scalable computation
- Review of related matrix algebra & calculus
Inverted Indexing

- Index each unique term as
  \[ \text{termID}: ((DID, weight), (DID, weight), \ldots, (DID, weight)) \]

- Equivalent to constructing a bipartite graph with sparse links

![Bipartite Graph](image)

1/17/2019 @Yiming Yang, Scalable Indexing

Cosine Similarity Computation

- Score a doc only if it shares words with the query;
- No need to compute \( \|q\| \) if we only want to rank doc's given \( q \).

\[
\cos(x_i, q) = \frac{\sum_{j, q_j \neq 0} q_j x_{ij}}{\|x_i\| \|q\|}
\]
Time/Space Savings

- n doc’s, m words in the vocabulary
- K unique words per doc on average
- K₉ unique words per query on average

Space Saving?  Time Saving in Retrieval?

\[
\text{DnsMtrx Space: } O(mn) \quad \text{DnsMtrx Time: } O(mn)
\]
\[
\text{InvIndx Space: } O(kn) \quad \text{InvIndx Time: } O\left(\frac{k \cdot kn}{m}\right)
\]
\[
\frac{\text{Space(dnsMtrx)}}{\text{Space(invIndx)}} = \frac{m}{k}
\]
\[
\frac{\text{Time(dnsMtrx)}}{\text{Time(invIndx)}} = \frac{m^2}{k \cdot k}
\]

Time/Space Saving Factors

<table>
<thead>
<tr>
<th>kq</th>
<th>k</th>
<th>m</th>
<th>space saving (m/k)</th>
<th>time saving (m^2/k*kq)</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>200</td>
<td>1,000</td>
<td>5</td>
<td>1,667</td>
</tr>
<tr>
<td>3</td>
<td>200</td>
<td>100,000</td>
<td>500</td>
<td>16,666,667</td>
</tr>
<tr>
<td>3</td>
<td>200</td>
<td>1,000,000</td>
<td>5,000</td>
<td>1,666,666,667</td>
</tr>
</tbody>
</table>

1/17/2019  @Yiming Yang, Scalable Indexing
When shall we use inverted indexing?

- Use it for computing $Xq$
  - If data matrix (X) is relatively stable, large and highly sparse
- Not suitable
  - If new documents arrive frequently (time series) and if updating the inverted index of matrix X for each new doc is too costly.
  - E.g., in filtering of news stories with fixed queries - re-index the entire document collection constantly (for every new doc) would be too costly.

Typical Text Mining Tasks

- Information Retrieval (search)
- Collaborative Filtering (CF)
- Link Analysis (HITS, PageRank)
- Text Categorization (TC)
- Dimensionality reduction (PCA, SVD, MF)
High-dimensional Sparse Data in Collaborative Filtering (CF)

Given a "query" (a new user) we find its k-nearest neighbor (kNN) users in the matrix ("training set") for predicting the "taste" of the new user.

\[ X_q = \begin{bmatrix} x_{11} & x_{12} & \cdots & x_{1m} \\ x_{21} & x_{22} & \cdots & x_{2m} \\ \vdots & \vdots & \ddots & \vdots \\ x_{n1} & x_{n2} & \cdots & x_{nm} \end{bmatrix} \]

rating (1~5) by user i on item j

Similarity scores of users w.r.t. q

\[ \begin{bmatrix} q_1 \\ q_2 \\ \vdots \\ q_m \end{bmatrix} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix} \]

Collaborative Filtering (CF)

Step 1. Find kNN(q)

\[ Xq \rightarrow 0 \rightarrow Xq \rightarrow y' \]

Step 2. Predict the new user's profile q'

\[ q' = \frac{1}{C} X^T y' = \frac{1}{C} \sum_{i \in \text{kNN}(q)} y_i \cdot x_i \quad (C \text{ is some normalization factor}) \]

Sparse matrix times a sparse vector
Typical Text Mining Tasks

- Information Retrieval (search)
- Collaborative Filtering (CF)
- Link Analysis (HITS, PageRank)
- Text Categorization (TC)
- Dimensionality reduction (PCA, SVD, MF)

High-dimensional Sparse Data in Link Analysis (HITS or PageRank)

Adjacency Matrix $A$ (in HITS)

<table>
<thead>
<tr>
<th></th>
<th>$x_1$</th>
<th>$x_2$</th>
<th>$x_3$</th>
<th>$x_4$</th>
<th>$x_5$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$y_1$</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>$y_2$</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>$y_3$</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>$y_4$</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>$y_5$</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

$A[i,j] = 1$ iff there is a link from $i$ to $j$.

Probabilistic Transition Matrix $T$ (in PageRank)

<table>
<thead>
<tr>
<th>$v_1$</th>
<th>$v_2$</th>
<th>$v_3$</th>
<th>$v_4$</th>
<th>$v_5$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0.5</td>
<td>0.5</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0.33</td>
<td>0</td>
<td>0.33</td>
<td>0</td>
<td>0.33</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

$T[i,j]$'s sum to 1 over $j$'s.

Power iteration is to compute $x := A^T A x$ repetitively.
Outline

✓ Big data and high-dimensional sparse matrices
✓ Inverted indexing for scalable computation
◼ Review of related matrix algebra & calculus

Vector and its norm

- $x \in \mathbb{R}^d$ is a vertical vector, e.g., $x = \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix}$.

- $\|x\|_p = p\sqrt{|x_1|^p + |x_2|^p + \cdots + |x_d|^p}$ is the $p$-norm of $x$ with $p \in [1, \infty)$.
  - Illustrated in the right graphs are the contours of $\|x\|_p = 1$ in $2$-D with different values of $p$. 
Common Operations

- Weighted sum of columns

\[Ax = \sum_{j=1}^{m} x_j A_j\]

- Weighted sum of rows

\[x^T A = \sum_{i=1}^{n} x_i A_i\]

Common Operations (cont’d)

- Weighted sum of columns

\[\Lambda A = \sum_{j=1}^{m} \lambda_j A_j\]

- Weighted sum of rows

\[\Lambda A = \sum_{i=1}^{n} \lambda_i A_i\]
Matrix-Vector Multiplication as a Linear Transformation

\[ x' = Ax \]

Eigenvalue & Eigenvector

• The linear transformation can only change the scale but not the direction of its eigenvector.

\[ Au = \lambda u \]
Common Operations (cont'd)

For $a, b \in R^m$

- Dotproduct $a \cdot b \triangleq \sum_{i=1}^{m} a_i b_i$
- Cosine $\cos(a, b) \triangleq \frac{a \cdot b}{\|a\| \times \|b\|}$
- Vector projection $b$ onto $a$

$$\|b\| \cos(\theta) = \|b\| \frac{a \cdot b}{\|a\| \times \|b\|} = \frac{a \cdot b}{\|a\|} = \frac{a}{\|a\|} \cdot b$$

Hyperplanes

$$h_1 : w^T x - b = 1$$
$$h_2 : w^T x - b = 0$$
$$h_3 : w^T x - b = -1$$

Which one is the best for separating red and black?

©Yiming Yang, Scalable Indexing 1/17/2019
Hyperplane Properties

Proof of $d = d(x, y, h) = \frac{y(w^T x - b)}{||w||}$

Case 1: $y = 1$, $x = z + d\frac{w}{||w||}$

Case 2: $y = -1$, $x = z - d\frac{w}{||w||}$, ...
Matrix Calculus

- Useful in optimizing prediction models
- Different types of derivatives
  - Scalar-by-scalar
  - Scalar-by-vector
  - Vector-by-vector
  - ...

Matrix Derivatives

- Scalar-by-scalar

\[ x \in R, \quad f(x) \in R \]

Example 1. \[ f(x) = ax^2 + b \]
\[ \frac{df}{dx} = 2ax, \quad \frac{d^2f}{dx^2} = 2a \]

Example 2. \[ f(x) = x^2 \log x, \quad \text{let } u = x^2, \quad v = \log x \]
\[ \frac{df}{dx} = u \frac{dv}{dx} + \frac{du}{dx} v = x^2 \frac{1}{x} + 2x \log x \]
Matrix Derivatives (cont’d)

- **Scalar-by-vector**
  
  \[ x \in \mathbb{R}^m, f \in \mathbb{R} \]

- The gradient
  
  \[ \nabla f \equiv \frac{\partial f}{\partial x} \equiv \left( \frac{\partial f}{\partial x_1}, \frac{\partial f}{\partial x_2}, \ldots, \frac{\partial f}{\partial x_m} \right) \in \mathbb{R}^m \]

- The Hessian
  
  \[ H \equiv \nabla \nabla f \equiv \left[ \frac{\partial^2 f}{\partial x_i \partial x_j} \right] \in \mathbb{R}^{m \times m}, \quad i, j \in \{1, \ldots, m\} \]

An example of scalar-by-vector

- e.g., \( x = (x_1, x_2)^T, \quad f(x) = ax_1^2 + bx_1x_2 \)

- The gradient
  
  \[ \nabla f = \left[ \frac{\partial f}{\partial x_1}, \frac{\partial f}{\partial x_2} \right] = \left[ 2ax_1 + bx_2, bx_1 \right] \]

- The Hessian
  
  \[ H \equiv \nabla \nabla f = \begin{bmatrix} 2a & b \\ b & 0 \end{bmatrix} \]
Matrix Derivatives (cont’d)

- **Vector-by-vector**
  
e.g. $x \in \mathbb{R}^m$, $y(x) = Ax \in \mathbb{R}^n$, $A \in \mathbb{R}^{n \times m}$
- The gradient
  
  $$\nabla y \equiv \left[ \frac{\partial y_j}{\partial x_i} \right] \in \mathbb{R}^{n \times m}, \quad \nabla y = A$$

Example

- $x \in \mathbb{R}^d$, $A \in \mathbb{R}^{d \times d}$, $f = x^T Ax$ $\leftarrow$ Scalar
  
  $\nabla f = ?$ $\leftarrow$ Vector (1-by-d)

  Let $u = x$, $v = Ax$, $f = u^T v$.
  
  $$\frac{\partial u}{\partial x} = I_{d \times d}, \quad \frac{\partial v}{\partial x} = A, \quad u^T \frac{\partial v}{\partial x} = x^T A, \quad v^T \frac{\partial u}{\partial x} = x^T A^T I$$

  $$u^T \frac{\partial v}{\partial x} + v^T \frac{\partial u}{\partial x} = x^T A + x^T A^T = x^T (A + A^T) \leftarrow \text{Vector (1-by-d)}$$