Approximate Matrix Multiplication for Pattern Recognition Tasks

Edith Cohen David D. Lewis

AT&T Labs – Research

http://www.research.att.com/~{edith,lewis}

{edith,lewis}@research.att.com

Outline

1. Linear Models in Pattern Recognition (background):
   - ◇ Background
   - ◇ Text retrieval example
   - ◇ Problem formulation
2. Approximate Matrix Multiplication
   - ◇ Results
   - ◇ Algorithmic ideas
3. Experimental Results
   - ◇ Sparse text-retrieval matrices
   - ◇ Dense real-valued matrices (LSI)

Linear Models in Information Retrieval

- Set of instances:
  Documents, Voice samples, Images, ...
- Every instance is represented by a numeric Feature Vector.
  Features may correspond to Words or Terms, Spectral coefficients, Center of mass of black pixels, ...
- The premise: Similarity of instances corresponds to Dot products, Cosines, or Euclidean distances between the feature vectors.
- Applications: Linear classifiers, Clustering, Similarity tests...
- Similarity problems: (using dot-product (cosine) similarity)
  - Online: For a query (represented as a feature vectors), find database instances with highest dot product with the query.
  - Offline: Given two collections of instances, identify, for each instance, most similar instances from the other collection. (applies to a collection and itself)

Nearest Neighbors for Euclidean Distances

- Euclidean distances are related to dot-products:

$||a - b|| = \sqrt{||a||^2 + ||b||^2 - 2||a \cdot b||}$.

Computing n NN queries for a database of n d-dimensional vectors:

Exact distances:

◇ Brute Force: $O(dn^2)$ using MM (faster using FMM).

ϵ-approximate distances:

Very low dimensions:

$O(n \log n + n \exp(d)\epsilon^{-d} \log n)$ [AMNSW95]

$O(n \log n + n \exp(d)\epsilon^{-(d-1)/2} \log n)$ [Clarkson94]

High dimensions:

- Random projections on $9 \log n\epsilon^{-2}$ dimensions [Frankl-Maehara88]:

$O(nd\epsilon^{-2} \log n + n^2\epsilon^{-2} \log n)$.

- Kleinberg97: $O(d^2n + n(n + d \log^3 n))$

Text Retrieval Example

terms/docs	frcst1	frcst2	nfl1	nfl2
sun	2	0	0	0
highs	3	3	0	0
lows	3	3	0	0
snow	0	2	0	1
touch down	0	0	10	2
Jerry Rice	0	0	7	0
49ers	0	0	4	1
Elway	0	0	0	6

Normalized Vectors

0.43	0	0	0
0.64	0.64	0	0
0.64	0.64	0	0
0	0.43	0	0.15
0	0	0.78	0.31
0	0	0.54	0
0	0	0.31	0.15
0	0	0	0.93

Similarity Matrix:

	frcst1	frcst2	nfl1	nfl2
frcst1	1	0.82	0	0
frcst2	0.82	1	0	0.06
nfl1	0	0	1	0.29
nfl2	0	0.06	0.29	1

Problem Formulation and Challenges

Instances are represented as feature vectors.
The instances database is represented by a matrix A.

Online: Database is processed once.
Queries q arrive online, represented as feature vectors.
- Find the highest matches for q (large entries in $q^T A$).

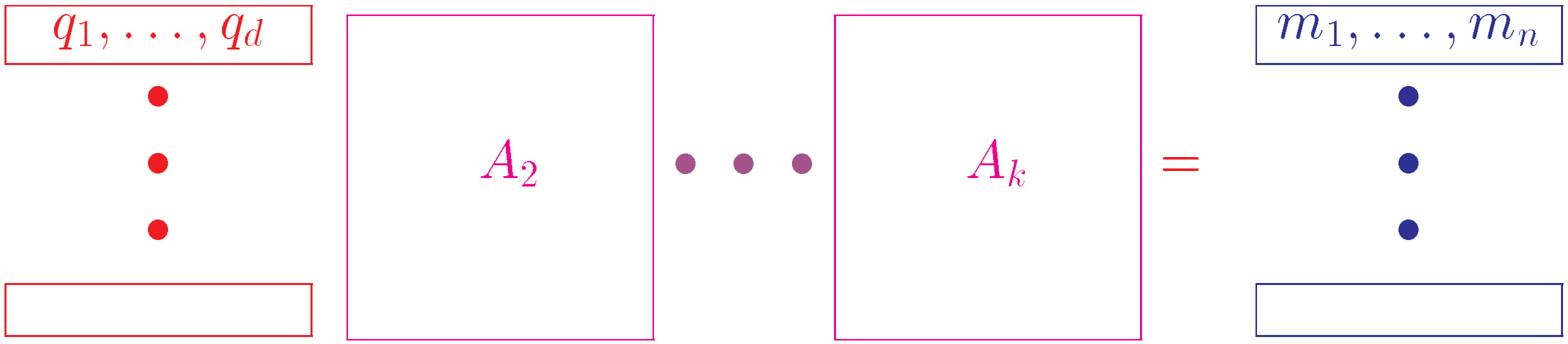
Offline: Given a collections of instances, find the highest matches in each row of $A^T A$. Given two collections A, B, find highest entries in each row of $A^T B$ (for every $b \in B$, the most similar instances in A.)

Obstacle: Matrix multiplication is slow.
◇ Dense product of k matrices takes $O(kn^3)$ time ($O(kn^{2.38})$/ FMM).
◇ Sparse matrices typically take $O(n \, \Sigma_{i=2}^{k} |A_i|)$.
There are existing heuristics for sparse data (compute until ordering of matches is determined, etc.) Not much was done for dense data.

Characteristics of Data:

- the highest matches, (relevant instances, nearest neighbors) are distinctly closer than the bulk of other points.
- Small differences in value are not meaningful: matches of values $m_i = \{0.71, 0.7\}$ are likely to be equally relevant.
- Large Scale:
  - ◇ Number of instances can be 10^4–10^7.
  - ◇ For sparse text-retrieval data number of terms may be 10^5.
  - ◇ For dense data (LSI), number of features is several hundreds.
- Common Proximity Measures are:
  cosine correlation, dot product, Euclidean
- Often nonnegative (doc-term matrices always are). Sometimes real-valued (LSI).

Approximate Matrix Multiplication



- $A_2, \ldots, A_k$ are preprocessed in linear time in #nonzeros
- For each q, the algorithm outputs $\hat{m}$ with expected value of m.

Theorem: Suppose all entries are nonnegative. Let $M = \Sigma_{i=1}^{n} m_i$. For any p, in time $\tilde{O}(kp^{-1}\epsilon^{-2})$, $\frac{m_i}{M} \geq p \Longrightarrow \frac{|\hat{m}_i - m_i|}{m_i} \leq \epsilon$

$\Rightarrow$ all entries over-the-mean $\frac{m_i}{M} \geq \frac{1}{n}$, are estimated in time $\tilde{O}(kn\epsilon^{-2})$.

(Exact computation takes $O(kn^2)$ for dense matrices.)

Product Estimation Algorithm

• Offline:
Input: matrices $A_1^{n_1\times n_2}, \ldots, A_k^{n_k\times n_{k+1}}$
Output: An estimate $\hat{P}^{n_1\times n_{k+1}}$ of the matrix product $P = A_1 \cdots A_k$.
• Online: $A_2, \ldots, A_k$ are provided at preprocessing. A_1 is supplied and P is estimated, row-by-row (Input:q Output:$\hat{m}$)

Preprocessing: Construct a directed $k+1$ layered graph from $A_1, \ldots, A_k$. Layer i has n_i nodes. Each 1st-layer node corresponds to a query q. The edges of the graph are labeled with probabilities.

Random walks from the node of q correspond to estimating
$m = q^T A_2 \cdots A_k$:
The walk terminates at node i in layer $k+1$ with probability $\frac{m_i}{M}$.
Each walk uses k steps and provides a sample.
When using S samples (walks), we obtain the estimate $\hat{m}_i = \frac{s_i}{S} M$, where s_i is the number of walks that terminated at node i.

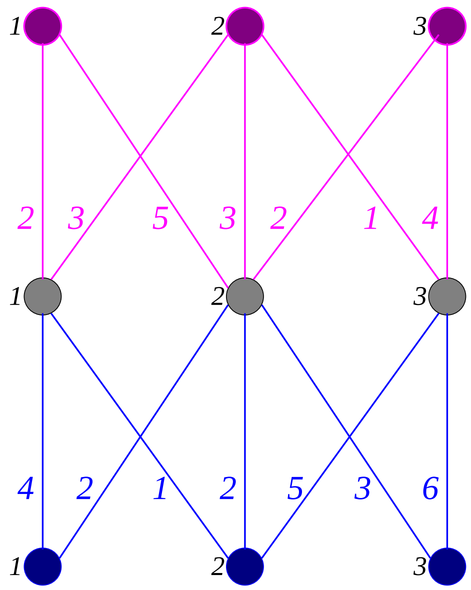
Example: estimating the product AB

matrices A, B

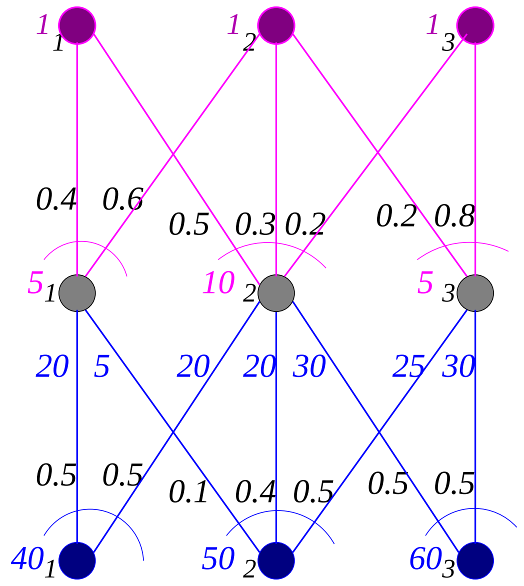
$$B = \begin{array}{|c|c|c|}\hline 2 & 3 & 0 \\ \hline 5 & 3 & 2 \\ \hline 0 & 1 & 4 \\ \hline\end{array}$$

$$A = \begin{array}{|c|c|c|}\hline 4 & 2 & 0 \\ \hline 1 & 2 & 5 \\ \hline 0 & 3 & 6 \\ \hline\end{array}$$

graph representation of AB



probabilities



$$AB = \begin{array}{|c|c|c|}\hline 18 & 18 & 4 \\ \hline 12 & 14 & 24 \\ \hline 15 & 15 & 30 \\ \hline\end{array}$$

$$\text{Row Sums} = \begin{array}{|c|}\hline 40 \\ \hline 50 \\ \hline 60 \\ \hline\end{array}$$

$$\text{Frac.} = \begin{array}{|c|c|c|}\hline 0.45 & 0.45 & 0.10 \\ \hline 0.24 & 0.28 & 0.48 \\ \hline 0.25 & 0.25 & 0.50 \\ \hline\end{array}$$

Estimating Products of Real Valued Matrices

Approach:

- Each product is composed of a positive and negative contributions. (both are nonnegative):

$$B_1 \cdots B_k = \overbrace{B_1 \cdots B_k}^{+} - \overbrace{B_1 \cdots B_k}^{-}$$

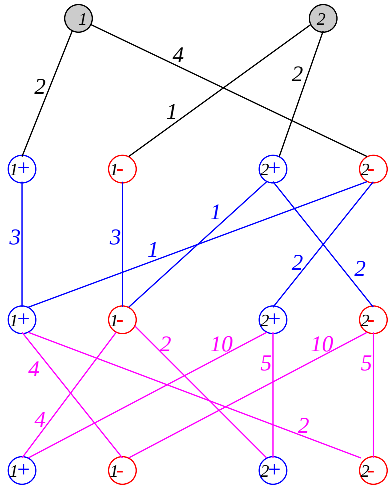
Each entry of $B_1 \cdots B_k$ is a sum of scalar k-products.
The positive contribution is the sum of all positive scalar products
The negative contribution is the (negated) sum of all negative scalar products

- each contribution is nonnegative, and can be estimated similarly to a nonnegative product.

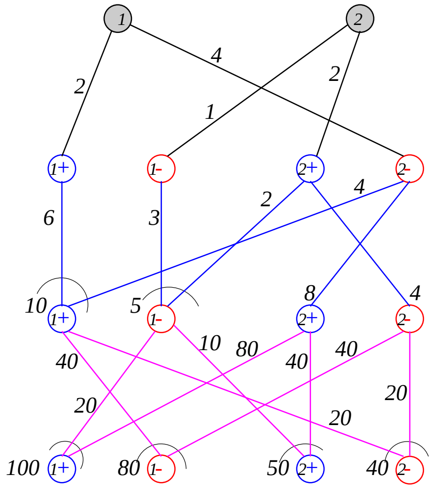
- Weaker estimate quality can be guaranteed.

Example of sampling for real valued data

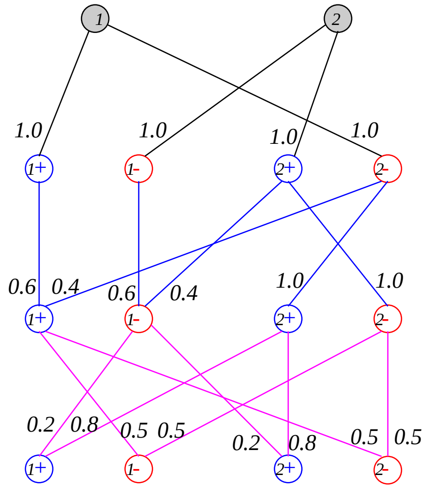
Graph representation of ABC



Sums



Transition probabilities



$$A = \begin{array}{|c|c|}\hline -4 & +10 \\ \hline -2 & +5 \\ \hline \end{array} \quad B = \begin{array}{|c|c|}\hline +3 & -1 \\ \hline 0 & -2 \\ \hline \end{array} \quad C = \begin{array}{|c|c|}\hline +2 & -1 \\ \hline -4 & +2 \\ \hline \end{array}$$

$$BC = \begin{array}{|c|c|}\hline 10 & -5 \\ \hline 8 & -4 \\ \hline \end{array} \quad \overset{+}{\overbrace{BC}} = \begin{array}{|c|c|}\hline 10 & 0 \\ \hline 8 & 0 \\ \hline \end{array} \quad \overset{-}{\overbrace{BC}} = \begin{array}{|c|c|}\hline 0 & 5 \\ \hline 0 & 4 \\ \hline \end{array}$$

$$ABC = \begin{array}{|c|c|}\hline +40 & -20 \\ \hline +20 & -10 \\ \hline \end{array} \quad \overset{+}{\overbrace{ABC}} = \begin{array}{|c|c|}\hline 80 & 20 \\ \hline 40 & 10 \\ \hline \end{array} \quad \overset{-}{\overbrace{ABC}} = \begin{array}{|c|c|}\hline 40 & 40 \\ \hline 20 & 20 \\ \hline \end{array}$$

Experiments

Nearest-Neighbors for Text-Retrieval data (collection of documents)

Representation: Matrix A. Every row of A corresponds to a document. Every document vector is normalized. Proximity measure is the cosine correlation.

1. Sparse nonnegative data:
   22K Reuters newswire articles.
   Using a sparse document-term representation. (106K terms)
2. Dense real-valued data:
   5000 Encyclopedia articles.
   Original representation: nonnegative sparse 5K $\times$ 76K matrix.
   with LSI: Dense matrix of dimension 5K $\times$ 320.

Task: Determine, for each document, its nearest neighbors.
That is, the largest entries in each row of AA^T.

Implementation and Algorithms

◇ Presampling: All samples are precomputed in a preprocessing step.

- For each feature, we produced a list of samples.
- For each query, we retrieved precomputed samples.

◇ Combined algorithm: for each query doc, we

1. used the sampling alg to isolate documents with high estimated matches.
2. computed exact dot-products (matches) for (small set of) docs with high estimates.

◇ For real-valued data: we

1. Transformed vectors to a positive form (high "transformed" dot-product correspond to high original dot-products)
2. Identified a set H of high "positive" matches
3. Computed exact dot-product values for H

Empirical Results

Dense real-valued data (5K documents)

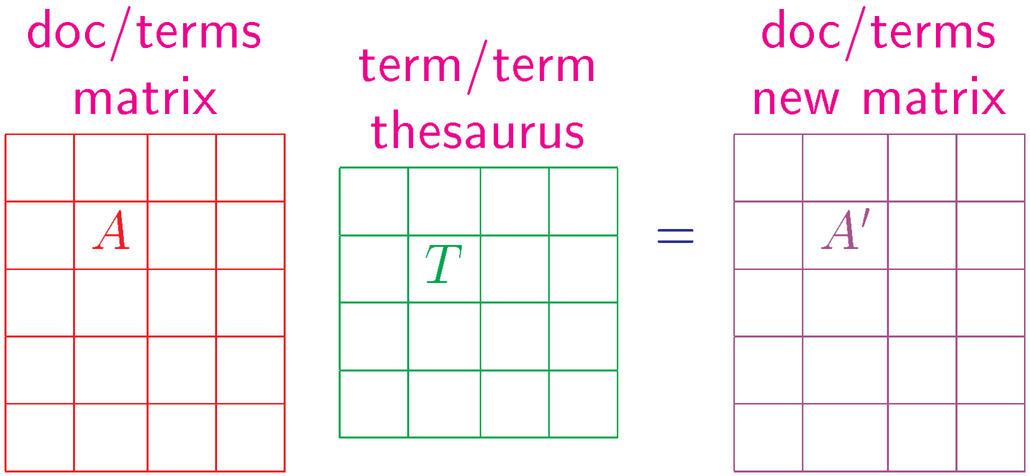
Alg	CPU time (sec)	Instances with Dot Product of at Least						
		0.55	0.60	0.65	0.70	0.75	0.80	0.85
F	3708	106368	68872	44412	28320	17688	11256	7568
D1	9%	85%	90%	94.2%	96.7%	98.4%	99.3%	99.8%
D2	18%	99%	99.6%	99.8%	99.9%	100%	100%	100%

Sparse nonnegative data (22K documents)

Alg	CPU time (sec)	Instances with Dot Product of at Least						
		0.15	0.20	0.25	0.30	0.35	0.40	0.45
F	2266	1755221	550871	243193	142323	97035	71969	54921
S1	17%	64.1%	88.5%	97.5%	99.5%	99.9%	100%	100%
S2	23%	89.1%	97.9%	99.7%	100%	100%	100%	100%

Text-retrieval Application of Estimating Chain Products (3 or more matrices)

Our algorithm and implementation apply to chain products



New doc/terms matrix $A' = AT$ "smoothes" term weights.

- A' is kept in product form (may require too much storage/compute time)
- The new similarity matrix is ATT^TA^T. The database matrix is AT (or ATT^T). The match values for a query q are ATq (or ATT^Tq).

Summary

- We introduced approximate multiplication algorithm of matrices. Our algorithm uses random sampling.

- For many tasks, (asymptotically and in practice) our algorithm is considerably faster than "exact" matrix multiplication

- Better estimation accuracy can be guaranteed for nonnegative data. In practice, our method is applicable to real-valued data.

- We demonstrated, empirically, significant speedups for text-retrieval tasks on both sparse nonnegative document-term matrices and dense real-valued LSI matrices.