Matrix Factorization
Outline

Matrix Factorization (collaborative filtering)

Sparse subspace embedding

Stochastic Gradient Descent (on the board)
Collaborative Filtering

Goal: predict users’ movie ratings based on past ratings of other movies

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th>1</th>
<th>?</th>
<th>?</th>
<th>4</th>
<th>5</th>
<th>?</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>?</td>
<td>5</td>
<td>?</td>
<td>?</td>
<td>?</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

$R = \begin{pmatrix}
1 & ? & ? & 4 & 5 & ? & 3 \\
\end{pmatrix}$
Model and Algorithm

Model $R$ as product of user and movie feature matrices $A$ and $B$ of size $U \times K$ and $M \times K$

$$R = A B^T$$

Alternating Least Squares (ALS)
- Start with random $A$ & $B$
- Optimize user vectors ($A$) based on movies
- Optimize movie vectors ($B$) based on users
- Repeat until converged
Alternating Least Squares

1. Start with random $A_1$, $B_1$
2. Solve for $A_2$ to minimize $\|R - A_2B_1^T\|$
3. Solve for $B_2$ to minimize $\|R - A_2B_2^T\|$
4. Repeat until convergence
Optimization problem

Iterate:

\[ f[i] = \arg \min_{w \in \mathbb{R}^d} \sum_{j \in \text{Nbrs}(i)} (r_{ij} - w^T f[j])^2 + \lambda \|w\|_2^2 \]
Attempt 1: Broadcast All

- Master loads (small) data file and initializes models.
- Master broadcasts data and initial models.
- At each iteration, updated models are broadcast again.
- Works OK for small data.
- Lots of communication overhead - doesn’t scale well.
Attempt 2: Data Parallel

- Workers load data
- Master broadcasts initial models
- At each iteration, updated models are broadcast again
- Much better scaling
- Works on large datasets
- Works well for smaller models. (low K)
Attempt 3: Fully Parallel

- Workers load data
- Models are instantiated at workers.
- At each iteration, models are shared via join between workers.
- Much better scalability.
- Works on large datasets
ALS on Spark

Cache 2 copies of $R$ in memory, one partitioned by rows and one by columns.

Keep $A$ & $B$ partitioned in corresponding way.

Operate on blocks to lower communication.
ALS Results

- Mahout / Hadoop: 4208 seconds
- Spark (Scala): 481 seconds
- GraphLab (C++): 297 seconds
Sparse subspace embedding
Sparse Subspace Embedding

\[ S = \Phi A \]  
(\( \Phi \) has one nonzero per column)

\[
\begin{align*}
A &= \# \text{ RDD of vectors, one per row} \\
s &= 10000 \quad \# \text{ embedding dimension} \\
S &= A.\text{map}(\lambda \text{ row}: (\text{randint}(1, s), \text{gauss}(0, 1) * \text{row}))) \backslash \\
&\quad .\text{reduceByKey}(\lambda a, b: a + b) \backslash \\
&\quad .\text{values}()
\end{align*}
\]

[Clarkson and Woodruff, STOC ’13]
Stochastic Gradient Descent (on the board, in the notes)