BUILDING A PREDICTIVE MODEL
AN EXAMPLE OF A PRODUCT RECOMMENDATION ENGINE

Alex Lin
Senior Architect
Intelligent Mining
alin@intelligentmining.com
Outline

- Predictive modeling methodology
- k-Nearest Neighbor (kNN) algorithm
- Singular value decomposition (SVD) method for dimensionality reduction
- Using a synthetic data set to test and improve your model
- Experiment and results
The Business Problem

- Design product recommender solution that will increase revenue.
How Do We Increase Revenue?

- Increase Revenue
- Increase Conversion
- Increase Avg. Order Value
- Increase Unit Price
- Increase Units / Order
Example

Is this recommendation effective?

- Increase Unit Price
- Increase Units / Order
What am I going to do?
Predictive Model

- **Framework**

  - Data
  - Features
  - ML Algorithm
  - Prediction Output

  - What data?
  - What feature?
  - Which Algorithm?
  - Cross-sell & Up-sell Recommendation
What Data to Use?

- Explicit data
  - Ratings
  - Comments

- Implicit data
  - Order history / Return history
  - Cart events
  - Page views
  - Click-thru
  - Search log

- In today’s talk we only use Order history and Cart events
Predictive Model

Data
Order History
Cart Events

Features
What feature?

ML Algorithm
Which Algorithm?

Prediction Output
Cross-sell & Up-sell Recommendation
What Features to Use?

- We know that a given product tends to get purchased by customers with similar tastes or needs.
- Use user engagement data to describe a product.

<table>
<thead>
<tr>
<th>item</th>
<th>users</th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>17</td>
<td>1</td>
<td>.25</td>
<td>.25</td>
<td>1</td>
<td>.25</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

user engagement vector
Data Representation / Features

- When we merge every item’s user engagement vector, we got a $m \times n$ item-user matrix

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>...</th>
<th>n</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>.25</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>.25</td>
<td></td>
<td></td>
<td>.25</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>.25</td>
<td></td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>.25</td>
<td></td>
<td>1</td>
<td></td>
<td>.25</td>
<td></td>
<td>.25</td>
<td></td>
<td>1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>m</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Data Normalization

- Ensure the magnitudes of the entries in the dataset matrix are appropriate

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>...</th>
<th>n</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>.5</td>
<td>.9</td>
<td>.92</td>
<td>.49</td>
<td>.2</td>
<td>.79</td>
<td>.73</td>
<td>.69</td>
<td>.52</td>
<td>.8</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>.67</td>
<td>.46</td>
<td>.73</td>
<td>.8</td>
<td>.82</td>
<td>.76</td>
<td>.69</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>.39</td>
<td>.82</td>
<td>.76</td>
<td>.69</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- Remove column average – so frequent buyers don’t dominate the model
Data Normalization

- Different engagement data points (Order / Cart / Page View) should have different weights
- Common normalization strategies:
  - Remove column average
  - Remove row average
  - Remove global mean
  - Z-score
  - Fill-in the null values
Predictive Model

Data

Order History
Cart Events

Features

User engagement
vector

ML Algorithm

Which Algorithm?

Prediction Output

Cross-sell & Up-sell Recommendation

Data Normalization

| users | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | ... | n |
|-------|---|---|---|---|---|---|---|---|---|----|     |    |
| 17    | 1 | .5| 1 | .5| 1 |    |    |    |    |     |     | n  |
Which Algorithm?

- How do we find the items that have similar user engagement data?

- We can find the items that have similar user engagement vectors with kNN algorithm.
**k-Nearest Neighbor (kNN)**

- Find the \( k \) items that have the most similar user engagement vectors

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>...</th>
<th>n</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>.5</td>
<td>1</td>
<td></td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td>.5</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td></td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td></td>
<td></td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>.5</td>
<td></td>
<td>1</td>
<td></td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>...</td>
<td>.5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>m</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td>.5</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- Nearest Neighbors of Item 4 = [2, 3, 1]
Similarity Measure for kNN

- **Jaccard coefficient:**
  \[ \text{sim}(a, b) = \frac{(1 + 1)}{(1 + 1 + 1) + (1 + 1 + 1 + 1) - (1 + 1)} \]

- **Cosine similarity:**
  \[ \text{sim}(a, b) = \cos(a, b) = \frac{a \cdot b}{\|a\|_2 \cdot \|b\|_2} = \frac{(1 \times 1 + 0.5 \times 1)}{\sqrt{(1^2 + 0.5^2 + 1^2) \times (1^2 + 0.5^2 + 1^2 + 1^2)}} \]

- **Pearson Correlation:**
  \[ \text{corr}(a, b) = \frac{\sum_i (r_{ai} - \overline{r}_a)(r_{bi} - \overline{r}_b)}{\sqrt{\sum_i (r_{ai} - \overline{r}_a)^2 \sum_i (r_{bi} - \overline{r}_b)^2}} = \frac{m \sum a_i b_i - \sum a_i \sum b_i}{\sqrt{m \sum a_i^2 - (\sum a_i)^2} \sqrt{m \sum b_i^2 - (\sum b_i)^2}} \]

\[= \frac{\text{match}_\text{cols} \cdot \text{Dotprod}(a, b) - \text{sum}(a) \cdot \text{sum}(b)}{\sqrt{\text{match}_\text{cols} \cdot \text{sum}(a^2) - (\text{sum}(a))^2} \sqrt{\text{match}_\text{cols} \cdot \text{sum}(b^2) - (\text{sum}(b))^2}} \]
k-Nearest Neighbor (kNN)

kNN $k=5$

Nearest Neighbors(8) = [9, 6, 3, 1, 2]
Predictive Model

- **Ver. 1: kNN**

  - **Data**
    - Order History
    - Cart Events
  - **Features**
    - User engagement vector
  - **ML Algorithm**
    - k-Nearest Neighbor (kNN)
  - **Prediction Output**
    - Cross-sell & Up-sell Recommendation

  ![Data Normalization](chart)
Cosine Similarity – Code fragment

```c
long i_cnt = 100000; // number of items 100K
long u_cnt = 2000000; // number of users 2M
double data[i_cnt][u_cnt]; // 100K by 2M dataset matrix (in reality, it needs to be malloc allocation)
double norm[i_cnt];

// assume data matrix is loaded
......
// calculate vector norm for each user engagement vector
for (i=0; i<i_cnt; i++) {
    norm[i] = 0;
    for (f=0; f<u_cnt; f++) {
        norm[i] += data[i][f] * data[i][f];
    }
    norm[i] = sqrt(norm[i]);
}

// cosine similarity calculation
for (i=0; i<i_cnt; i++) {
    for (j=0; j<i_cnt; j++) {
        dot_product = 0;
        for (f=0; f<u_cnt; f++) {
            dot_product += data[i][f] * data[j][f];
        }
        printf("%d %d %lf
", i, j, dot_product/(norm[i] * norm[j]));
    }
}
```

1. 100K rows x 100K rows x 2M features --> scalability problem
   - kd-tree, Locality sensitive hashing,
   - MapReduce/Hadoop, Multicore/Threading, Stream Processors

2. data[i] is high-dimensional and sparse, similarity measures
   are not reliable --> accuracy problem
   This leads us to The SVD dimensionality reduction!
Singular Value Decomposition (SVD)

\[ A = U \times S \times V^T \]

- Low rank approx. Item profile is \( U_k \times \sqrt{S_k} \)
- Low rank approx. User profile is \( \sqrt{S_k} \times V_k^T \)
- Low rank approx. Item-User matrix is \( U_k \times \sqrt{S_k} \times \sqrt{S_k} \times V_k^T \)
Reduced SVD

\[ A_k = U_k \times S_k \times V_k^T \]

- Low rank approx. Item profile is \( U_k \times \sqrt{S_k} \)

- Diagram showing matrices:
  - \( A_k \): 100K x 2M matrix
  - \( U_k \): 100K x 3 matrix
  - \( S_k \): 3 x 3 matrix with values
    - 7 0 0
    - 0 3 0
    - 0 0 1
  - \( V_k^T \): 3 x 2M matrix
SVD Factor Interpretation

- Singular values plot (rank=512)

**Singular Values**

<table>
<thead>
<tr>
<th>7</th>
<th>0</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>3</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

3 x 3 matrix

Descending Singular Values

More Significant  Latent Factors  Noises + Others  Less Significant
SVD Dimensionality Reduction

Need to find the most optimal low rank !!

\( U_k \cdot \sqrt{S_k} \)

<----- latent factors ----->

# of users

items

Need to find the most optimal low rank !!

rank

3

10

100k x 3 matrix

24
Missing values

- Difference between “0” and “unknown”
- Missing values do NOT appear randomly.
- Value = (Preference Factors) + (Availability) – (Purchased elsewhere) – (Navigation inefficiency) – etc.
- Approx. Value = (Preference Factors) +/- (Noise)
- Modeling missing values correctly will help us make good recommendations, especially when working with an extremely sparse data set
Singular Value Decomposition (SVD)

- Use SVD to reduce dimensionality, so neighborhood formation happens in reduced user space
- SVD helps model to find the low rank approx. dataset matrix, while retaining the critical latent factors and ignoring noise.
- Optimal low rank needs to be tuned
- SVD is computationally expensive

SVD Libraries:
- Matlab: \([U, S, V] = \text{svds}(A,256);\)
- GHAPACK: [http://www.dcs.shef.ac.uk/~genevieve/ml.html](http://www.dcs.shef.ac.uk/~genevieve/ml.html)
Predictive Model

- **Ver. 2: SVD+kNN**

  ![Diagram of Predictive Model]

  - **Data**
    - Order History
    - Cart Events
  - **Features**
    - User engagement vector
    - Data Normalization
  - **ML Algorithm**
    - k-Nearest Neighbors (kNN) in reduced space
  - **Prediction Output**
    - Cross-sell & Up-sell Recommendation
Synthetic Data Set

- Why do we use synthetic data set?

- So we can test our new model in a controlled environment
Synthetic Data Set

- 16 latent factors synthetic e-commerce data set
  - Dimension: 1,000 (items) by 20,000 (users)
  - 16 user preference factors
  - 16 item property factors (non-negative)
  - Txn Set: \( n = 55,360 \), sparsity = 99.72%
  - Txn+Cart Set: \( n = 192,985 \), sparsity = 99.03%

<table>
<thead>
<tr>
<th>user_id</th>
<th>item_id</th>
<th>type</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>42</td>
<td>0.25</td>
</tr>
<tr>
<td>10</td>
<td>997</td>
<td>0.25</td>
</tr>
<tr>
<td>10</td>
<td>950</td>
<td>0.25</td>
</tr>
<tr>
<td>11</td>
<td>836</td>
<td>0.25</td>
</tr>
<tr>
<td>11</td>
<td>225</td>
<td>1</td>
</tr>
</tbody>
</table>
### Synthetic Data Set

#### Item property factors

1K x 16 matrix

<table>
<thead>
<tr>
<th>a</th>
<th>b</th>
<th>c</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

#### User preference factors

16 x 20K matrix

| x |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |

#### Purchase Likelihood score

1K x 20K matrix

<table>
<thead>
<tr>
<th>X₁₁</th>
<th>X₁₂</th>
<th>X₁₃</th>
<th>X₁₄</th>
<th>X₁₅</th>
<th>X₁₆</th>
</tr>
</thead>
<tbody>
<tr>
<td>X₂₁</td>
<td>X₂₂</td>
<td>X₂₃</td>
<td>X₂₄</td>
<td>X₂₅</td>
<td>X₂₆</td>
</tr>
<tr>
<td>X₃₁</td>
<td>X₃₂</td>
<td>X₃₃</td>
<td>X₃₄</td>
<td>X₃₅</td>
<td>X₃₆</td>
</tr>
<tr>
<td>X₄₁</td>
<td>X₄₂</td>
<td>X₄₃</td>
<td>X₄₄</td>
<td>X₄₅</td>
<td>X₄₆</td>
</tr>
<tr>
<td>X₅₁</td>
<td>X₅₂</td>
<td>X₅₃</td>
<td>X₅₄</td>
<td>X₅₅</td>
<td>X₅₆</td>
</tr>
</tbody>
</table>

#### Equations

\[ X_{32} = (a, b, c) \cdot (x, y, z) = a \cdot x + b \cdot y + c \cdot z \]

\[ X_{32} = \text{Likelihood of Item 3 being purchased by User 2} \]
Synthetic Data Set

- User 1 purchased Item 4 and Item 1

Sort by Purchase likelihood Score

Based on the distribution, pre-determine # of items purchased by an user (# of item=2)

From the top, select and skip certain items to create data sparsity.
Experiment Setup

- Each model (Random / kNN / SVD+kNN) will generate top 20 recommendations for each item.
- Compare model output to the actual top 20 provided by synthetic data set

Evaluation Metrics:

- Precision %: Overlapping of the top 20 between model output and actual (higher the better)
  \[
  \text{Precision} = \frac{|\{\text{Found}_{-}\text{Top20}_{-}\text{items}\} \cap \{\text{Actual}_{-}\text{Top20}_{-}\text{items}\}|}{|\{\text{Found}_{-}\text{Top20}_{-}\text{items}\}|}
  \]

- Quality metric: Average of the actual ranking in the model output (lower the better)

<table>
<thead>
<tr>
<th>1</th>
<th>2</th>
<th>30</th>
<th>47</th>
<th>50</th>
<th>21</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>368</td>
<td>62</td>
<td>900</td>
<td>510</td>
</tr>
</tbody>
</table>
Experimental Result

- **kNN vs. Random (Control)**

  - Precision % (higher is better)
  - Quality (Lower is better)
### Experimental Result

- **Precision % of SVD+kNN**

  ![Graph showing the precision of SVD+kNN for different SVD ranks](image)

  - **Recall %** (higher is better)
  - **Improvement**

  - **SVD Rank**
Experimental Result

- Quality of SVD+kNN

Quality (Lower is better)

Improvement

SVD Rank
Experimental Result

- The effect of using Cart data

Precision %
(higher is better)
Experimental Result

- The effect of using Cart data

Quality (Lower is better)
Outline

- Predictive modeling methodology
- k-Nearest Neighbor (kNN) algorithm
- Singular value decomposition (SVD) method for dimensionality reduction
- Using a synthetic data set to test and improve your model
- Experiment and results
References

- B. Sarwar, G. Karypis, J. Konstan, and J. Riedl "Application of Dimensionality Reduction in Recommender System A Case Study" In ACM WebKDD 2000 Web Mining for E-Commerce Workshop
- Apache Lucene Mahout  http://lucene.apache.org/mahout/
- Cofi: A Java-Based Collaborative Filtering Library  http://www.nongnu.org/cofi/
Thank you

- Any question or comment?