Recommender Systems: Content-based Systems & Collaborative Filtering
High Dimensional Data

- **High dim. data**
  - Locality sensitive hashing
  - Clustering
  - Dimensional reduction

- **Graph data**
  - Community Detection
  - Spam Detection

- **Infinite data**
  - Filtering streams
  - Web advertising
  - Queries on streams

- **Machine learning**
  - Decision Trees
  - Perceptron, kNN

- **Apps**
  - Recommender systems
  - Association Rules
  - Duplicate document detection
Example: Recommender Systems

- **Customer X**
  - Buys Metallica CD
  - Buys Megadeth CD

- **Customer Y**
  - Does search on Metallica
  - Recommender system suggests Megadeth from data collected about customer X
Recommendations

Examples:

- Amazon.com
- Pandora
- StumbleUpon
- del.icio.us
- Netflix
- Movielens
- Last.fm
- Google News
- YouTube
- Xbox Live

Search → Recommendations

Items

Products, web sites, blogs, news items, …
Shelf space is a scarce commodity for traditional retailers
- Also: TV networks, movie theaters,...

Web enables near-zero-cost dissemination of information about products
- From scarcity to abundance

More choice necessitates better filters
- Recommendation engines
- How Into Thin Air made Touching the Void a bestseller: http://www.wired.com/wired/archive/12.10/tail.html
Sidenote: The Long Tail

THE DOCUMENTARY NICHE GETS RICHER

More than 40,000 documentaries have been released, according to the Internet Movie Database. Of those, Amazon.com carries 40 percent, Netflix stocks 3 percent, and the average Blockbuster just .2 percent.

Sources: Erik Brynjolfsson and Jaffray Hu, MIT, and Michael Smith, Carnegie Mellon; Barnes & Noble; Netflix; RealNetworks. Source: Chris Anderson (2004)
Physical vs. Online

Read http://www.wired.com/wired/archive/12.10/tail.html to learn more!
Types of Recommendations

- **Editorial and hand curated**
  - List of favorites
  - Lists of “essential” items

- **Simple aggregates**
  - Top 10, Most Popular, Recent Uploads

- **Tailored to individual users**
  - Amazon, Netflix, ...
**Formal Model**

- \( X = \text{set of Customers} \)
- \( S = \text{set of Items} \)

**Utility function** \( u: X \times S \rightarrow R \)

- \( R = \text{set of ratings} \)
- \( R \) is a totally ordered set
- e.g., 0-5 stars, real number in \([0,1]\)
## Utility Matrix

<table>
<thead>
<tr>
<th></th>
<th>Avatar</th>
<th>LOTR</th>
<th>Matrix</th>
<th>Pirates</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alice</td>
<td>1</td>
<td>0.2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Bob</td>
<td>0.5</td>
<td>0.3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Carol</td>
<td>0.2</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>David</td>
<td></td>
<td></td>
<td></td>
<td>0.4</td>
</tr>
</tbody>
</table>
Key Problems

- (1) Gathering “known” ratings for matrix
  - How to collect the data in the utility matrix

- (2) Extrapolate unknown ratings from the known ones
  - Mainly interested in high unknown ratings
    - We are not interested in knowing what you don’t like but what you like

- (3) Evaluating extrapolation methods
  - How to measure success/performance of recommendation methods
(1) Gathering Ratings

- **Explicit**
  - Ask people to rate items
  - Doesn’t work well in practice – people can’t be bothered
  - Crowdsourcing: Pay people to label items

- **Implicit**
  - Learn ratings from user actions
    - E.g., purchase implies high rating
  - What about low ratings?
Key problem: Utility matrix $U$ is sparse

- Most people have not rated most items
- Cold start:
  - New items have no ratings
  - New users have no history

Three approaches to recommender systems:

1) Content-based
2) Collaborative
3) Latent factor based
Content-based Recommender Systems
Content-based Recommendations

- **Main idea:** Recommend items to customer \( x \) similar to previous items rated highly by \( x \)

**Example:**
- **Movie recommendations**
  - Recommend movies with same actor(s), director, genre, ...
- **Websites, blogs, news**
  - Recommend other sites with “similar” content
Plan of Action

Item profiles

User profile

match

likes

build

recommend

Red Circles Triangles
For each item, create an **item profile**

Profile is a set (vector) of features

- **Movies**: author, title, actor, director, ...
- **Text**: Set of “important” words in document

**How to pick important features?**

- Usual heuristic from text mining is **TF-IDF**
  (Term frequency * Inverse Doc Frequency)
  - Term ... Feature
  - Document ... Item
\( f_{ij} = \) frequency of term (feature) \( i \) in doc (item) \( j \)

\[
TF_{ij} = \frac{f_{ij}}{\max_k f_{kj}}
\]

\( n_i = \) number of docs that mention term \( i \)

\( N = \) total number of docs

\[
IDF_i = \log \frac{N}{n_i}
\]

TF-IDF score: \( w_{ij} = TF_{ij} \times IDF_i \)

**Doc profile** = set of words with highest TF-IDF scores, together with their scores

**Note:** we normalize TF to discount for “longer” documents
User Profiles and Prediction

- **User profile possibilities:**
  - Weighted average of rated item profiles
  - **Variation:** weight by difference from average rating for item
  - ...

- **Prediction heuristic:**
  - Given user profile $x$ and item profile $i$, estimate
    \[ u(x, i) = \cos(x, i) = \frac{x \cdot i}{||x|| \cdot ||i||} \]

Note: cosine similarity $u(x, i)$ defined this way ranges $[-1, 1]$, to get it to $[0, 1]$ we take $\text{arc cos } u(x,i)$. 
Pros: Content-based Approach

- +: No need for data on other users
  - No cold-start or sparsity problems
- +: Able to recommend to users with unique tastes
- +: Able to recommend new & unpopular items
  - No first-rater problem
- +: Able to provide explanations
  - Can provide explanations of recommended items by listing content-features that caused an item to be recommended
Cons: Content-based Approach

- Finding the appropriate features is hard
  - E.g., images, movies, music
- Recommendations for new users
  - How to build a user profile?
- Overspecialization
  - Never recommends items outside user’s content profile
  - People might have multiple interests
  - Unable to exploit quality judgments of other users
Collaborative Filtering

Harnessing quality judgments of other users
Collaborative Filtering

- Consider user $x$
- Find set $N$ of other users whose ratings are “similar” to $x$’s ratings
- Estimate $x$’s ratings based on ratings of users in $N$
Finding “Similar” Users

- Let \( r_x \) be the vector of user \( x \)’s ratings
- Jaccard similarity measure:
  - Problem: Ignores the value of the rating
- Cosine similarity measure:
  - \( \text{sim}(x, y) = \cos(r_x, r_y) = \frac{r_x \cdot r_y}{||r_x|| \cdot ||r_y||} \)
  - Problem: Treats some missing ratings as “negative”
- Pearson correlation coefficient:
  - \( S_{xy} \) = items rated by both users \( x \) and \( y \)
  - \( \text{sim}(x, y) = \frac{\sum_{s \in S_{xy}} (r_{xs} - \overline{r}_x)(r_{ys} - \overline{r}_y)}{\sqrt{\sum_{s \in S_{xy}} (r_{xs} - \overline{r}_x)^2} \sqrt{\sum_{s \in S_{xy}} (r_{ys} - \overline{r}_y)^2}} \)

\( r_x = [1, _, _, 1, 3] \)
\( r_y = [1, _, 2, 2, _] \)

\( r_x, r_y \) as sets:
\( r_x = \{1, 4, 5\} \)
\( r_y = \{1, 3, 4\} \)

\( r_x, r_y \) as points:
\( r_x = \{1, 0, 0, 1, 3\} \)
\( r_y = \{1, 0, 2, 2, 0\} \)
Intuitively we want: \( \text{sim}(A, B) > \text{sim}(A, C) \)

Jaccard similarity: \( \frac{1}{5} < \frac{2}{4} \)

Cosine similarity: \( 0.386 > 0.322 \)

- Considers missing ratings as “negative”
- **Solution: subtract the (row) mean**

\[
\text{sim}(x, y) = \frac{\sum_i r_{xi} \cdot r_{yi}}{\sqrt{\sum_i r_{xi}^2} \cdot \sqrt{\sum_i r_{yi}^2}}
\]

<table>
<thead>
<tr>
<th></th>
<th>HP1</th>
<th>HP2</th>
<th>HP3</th>
<th>TW</th>
<th>SW1</th>
<th>SW2</th>
<th>SW3</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>4</td>
<td>5</td>
<td></td>
<td>5</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>B</td>
<td>5</td>
<td>5</td>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C</td>
<td></td>
<td>2</td>
<td>4</td>
<td>5</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>D</td>
<td></td>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>3</td>
</tr>
</tbody>
</table>

\( \text{sim} A, B \) vs. \( A, C \): \( 0.092 > -0.559 \)

Notice cosine sim. is correlation when data is centered at 0
From similarity metric to recommendations:

- Let $r_x$ be the vector of user $x$’s ratings
- Let $N$ be the set of $k$ users most similar to $x$ who have rated item $i$

Prediction for item $s$ of user $x$:

- $r_{xi} = \frac{1}{k} \sum_{y \in N} r_{yi}$
- $r_{xi} = \frac{\sum_{y \in N} s_{xy} \cdot r_{yi}}{\sum_{y \in N} s_{xy}}$

Shorthand:

$s_{xy} = \text{sim}(x, y)$

Other options?

Many other tricks possible...
Item-Item Collaborative Filtering

- So far: User-user collaborative filtering
- Another view: Item-item
  - For item $i$, find other similar items
  - Estimate rating for item $i$ based on ratings for similar items
  - Can use same similarity metrics and prediction functions as in user-user model

\[
\hat{r}_{xi} = \frac{\sum_{j \in N(i;x)} s_{ij} \cdot r_{xj}}{\sum_{j \in N(i;x)} s_{ij}}
\]

$s_{ij}$... similarity of items $i$ and $j$
$r_{xj}$...rating of user $u$ on item $j$
$N(i;x)$... items similar to $i$ and rated by user $x$
<table>
<thead>
<tr>
<th>users</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>11</th>
<th>12</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>3</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>5</td>
<td>4</td>
<td>4</td>
<td>2</td>
<td>1</td>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>4</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>3</td>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>2</td>
<td>4</td>
<td>5</td>
<td>4</td>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>4</td>
<td>3</td>
<td>4</td>
<td>2</td>
<td>2</td>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>1</td>
<td>3</td>
<td>3</td>
<td>2</td>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- unknown rating
- rating between 1 to 5

Jure Leskovec, Stanford C246: Mining Massive Datasets
**Item-Item CF (|N|=2)**

<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>11</th>
<th>12</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>3</td>
<td>?</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td>5</td>
<td>4</td>
<td></td>
<td>4</td>
<td>2</td>
<td>1</td>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>4</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>3</td>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>2</td>
<td>4</td>
<td>5</td>
<td>4</td>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>4</td>
<td>3</td>
<td>4</td>
<td>2</td>
<td></td>
<td>2</td>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>1</td>
<td>3</td>
<td>3</td>
<td>2</td>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- estimate rating of movie 1 by user 5

1/25/2016

Jure Leskovec, Stanford C246: Mining Massive Datasets
### Item-Item CF ($|N|=2$)

#### Neighbor selection:
Identify movies similar to movie 1, rated by user 5

Here we use Pearson correlation as similarity:

1. Subtract mean rating $m_i$ from each movie $i$:
   
   $$m_1 = (1+3+5+5+4)/5 = 3.6$$

2. Compute cosine similarities between rows:

#### 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>11</th>
<th>12</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>3</td>
<td>?</td>
<td>5</td>
<td></td>
<td>5</td>
<td></td>
<td>5</td>
<td>4</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></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>4</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>3</td>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</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>5</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>6</td>
<td>1</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>2</td>
<td>2</td>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

#### Similarities

- $\text{sim}(1,m)$
  
  - 1.00
  - -0.18
  - 0.41
  - -0.10
  - -0.31
  - 0.59

1/25/2016

Jure Leskovec, Stanford C246: Mining Massive Datasets
## Item-Item CF ($|N|=2$)

### Compute similarity weights:

$s_{1,3}=0.41$, $s_{1,6}=0.59$
### Item-Item CF ($|N|=2$)

**Predict by taking weighted average:**

\[
r_{1.5} = \frac{(0.41 \times 2 + 0.59 \times 3)}{(0.41 + 0.59)} = 2.6
\]

![Matrix for Item-Item CF (|N|=2)](image)
CF: Common Practice

- Define similarity $s_{ij}$ of items $i$ and $j$
- Select $k$ nearest neighbors $N(i; x)$
  - Items most similar to $i$, that were rated by user $x$
- Estimate rating $r_{xi}$ as the weighted average:

$$r_{xi} = b_{xi} + \frac{\sum_{j \in N(i; x)} s_{ij} \cdot (r_{xj} - b_{xj})}{\sum_{j \in N(i; x)} s_{ij}}$$

baseline estimate for $r_{xi}$

$$b_{xi} = \mu + b_x + b_i$$

- $\mu$ = overall mean movie rating
- $b_x$ = rating deviation of user $x$
  - $(\text{avg. rating of user } x) - \mu$
- $b_i$ = rating deviation of movie $i$
## Item-Item vs. User-User

<table>
<thead>
<tr>
<th></th>
<th>Avatar</th>
<th>LOTR</th>
<th>Matrix</th>
<th>Pirates</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alice</td>
<td>1</td>
<td>0.8</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Bob</td>
<td>0.5</td>
<td>0.3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Carol</td>
<td>0.9</td>
<td>1</td>
<td>0.8</td>
<td></td>
</tr>
<tr>
<td>David</td>
<td>1</td>
<td>0.4</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- **In practice, it has been observed that item-item often works better than user-user**
- **Why?** Items are simpler, users have multiple tastes
Pros/Cons of Collaborative Filtering

+ Works for any kind of item
  - No feature selection needed

- Cold Start:
  - Need enough users in the system to find a match

- Sparsity:
  - The user/ratings matrix is sparse
  - Hard to find users that have rated the same items

- First rater:
  - Cannot recommend an item that has not been previously rated
  - New items, Esoteric items

- Popularity bias:
  - Cannot recommend items to someone with unique taste
  - Tends to recommend popular items
Hybrid Methods

- Implement two or more different recommenders and combine predictions
  - Perhaps using a linear model

- Add content-based methods to collaborative filtering
  - Item profiles for new item problem
  - Demographics to deal with new user problem
Remarks & Practical Tips

- Evaluation
- Error metrics
- Complexity / Speed
## Evaluation

### Movie Rating Matrix

<table>
<thead>
<tr>
<th></th>
<th>movies</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td>3</td>
<td>5</td>
</tr>
<tr>
<td>4</td>
<td>5</td>
</tr>
<tr>
<td>3</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td></td>
</tr>
</tbody>
</table>

### Users

<p>| |</p>
<table>
<thead>
<tr>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
</tbody>
</table>

### Details

- **Movies**: 5
- **Users**: 5
- **Rating Scale**: 5

1/25/2016

Jure Leskovec, Stanford C246: Mining Massive Datasets
## Evaluation

<table>
<thead>
<tr>
<th></th>
<th>movies</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1 3 4</td>
</tr>
<tr>
<td></td>
<td>3 5 5</td>
</tr>
<tr>
<td></td>
<td>4 5 5</td>
</tr>
<tr>
<td></td>
<td>3</td>
</tr>
<tr>
<td></td>
<td>3</td>
</tr>
<tr>
<td>users</td>
<td>2 ?</td>
</tr>
<tr>
<td></td>
<td>2 1</td>
</tr>
<tr>
<td></td>
<td>3</td>
</tr>
<tr>
<td></td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>1</td>
</tr>
</tbody>
</table>

**Test Data Set**
Evaluating Predictions

- **Compare predictions with known ratings**
  - **Root-mean-square error** (RMSE)
    \[ \sqrt{\sum_{xi} (r_{xi} - r^*_{xi})^2} \]
    where \( r_{xi} \) is predicted, \( r^*_{xi} \) is the true rating of \( x \) on \( i \)
  - **Precision at top 10:**
    \[ \% \text{ of those in top 10} \]
  - **Rank Correlation:**
    - Spearman’s correlation between system’s and user’s complete rankings

- **Another approach: 0/1 model**
  - **Coverage:**
    \[ \text{Number of items/users for which system can make predictions} \]
  - **Precision:**
    - Accuracy of predictions
  - **Receiver operating characteristic** (ROC)
    - Tradeoff curve between false positives and false negatives
Problems with Error Measures

- Narrow focus on accuracy sometimes misses the point
  - Prediction Diversity
  - Prediction Context
  - Order of predictions

- In practice, we care only to predict high ratings:
  - RMSE might penalize a method that does well for high ratings and badly for others
Expensive step is finding $k$ most similar customers: $O(|X|)$

Too expensive to do at runtime
- Could pre-compute
- Naïve pre-computation takes time $O(k \cdot |X|)$
  - $X$ ... set of customers

We already know how to do this!
- Near-neighbor search in high dimensions ($\text{LSH}$)
- Clustering
- Dimensionality reduction
Tip: Add Data

- Leverage all the data
  - Don’t try to reduce data size in an effort to make fancy algorithms work
  - Simple methods on large data do best

- Add more data
  - e.g., add IMDB data on genres

- More data beats better algorithms

http://anand.typepad.com/datawocky/2008/03/more-data-usual.html
On Thursday:
The Netflix prize and the Latent Factor Models
On Thursday: The Netflix Prize

- **Training data**
  - 100 million ratings, 480,000 users, 17,770 movies
  - 6 years of data: 2000-2005

- **Test data**
  - Last few ratings of each user (2.8 million)
  - Evaluation criterion: root mean squared error (RMSE)
  - Netflix Cinematch RMSE: 0.9514

- **Competition**
  - 2700+ teams
  - $1 million prize for 10% improvement on Cinematch
On Thursday: Latent Factor Models

- **Next topic:** Recommendations via Latent Factor models

The bubbles above represent products sized by sales volume. Products close to each other are recommended to each other.
Latent Factor Models (i.e., SVD++)

Geared towards females:
- The Princess Diaries
- Amadeus
- The Color Purple

Sense and Sensibility:
- The Lion King

Serious:
- Lethal Weapon
- Braveheart

Escapist:
- Ocean’s 11
- Dumb and Dumber
Figure 1: Singular Value Decomposition (SVD) of a user-movie matrix. The figure shows the top 20 movies with the highest score on the first and second factor vectors. The movies are plotted in the factor space, with the first factor vector on the x-axis and the second factor vector on the y-axis. The titles of some movies are included:

- Freddy Got Fingered
- Half Baked
- Natural Born Killers
- Scarse
- Julien Donkey-Boy
- Kill Bill: Vol. 1
- I Heart Huckabees
- Punch-Drunk Love
- The Royal Tenenbaums
- Being John Malkovich
- Lost in Translation
- Belle de Jour
- Citizen Kane
- Annie Hall
- Freddy vs. Jason
- Road Trip
- The Longest Yard
- The Fast and the Furious
- Armageddon
- Catwoman
- Coyote Ugly
- The Wizard of Oz
- Maid in Manhattan
- Runaway Bride
- Stepmom
- Sister Act
- The Way We Were
- The Sound of Music
- The Waltons: Season 1
- Moonstruck
- Sophie's Choice
- The Sound of Music
- The Waltons: Season 1