Case Study 4: Collaborative Filtering

Probabilistic Matrix Factorization

Machine Learning for Big Data
CSE547/STAT548, University of Washington
Emily Fox
May 19th, 2015

Matrix Completion Problem

$X_{ij}$ known for black cells
$X_{ij}$ unknown for white cells
Rows index users
Columns index movies

• Filling missing data?
Coordinate Descent for Matrix Factorization: Alternating Least-Squares

\[
\min_{L,R} \sum_{(u,v): r_{uv} \neq ?} (L_u \cdot R_v - r_{uv})^2 + \lambda_u \|L\| + \lambda_v \|R\|
\]

- **Fix movie factors, optimize for user factors**
  - Independent least-squares over users
  \[
  \min_{L_u} \sum_{v \in V_u} (L_u \cdot R_v - r_{uv})^2 + \lambda_u \|L\|
  \]

- **Fix user factors, optimize for movie factors**
  - Independent least-squares over movies
  \[
  \min_{R_v} \sum_{u \in U_v} (L_u \cdot R_v - r_{uv})^2 + \lambda_v \|R\|
  \]

- System may be underdetermined: use regularization

- Converges to local optima

Proxabilistic Matrix Factorization (PMF)

- A generative process:
  - Pick user \( u \) factors
  - Pick movie \( v \) factors
  - For each \((u,v)\) pair observed:
    - Pick rating as \( L_u \cdot R_v + \text{noise} \)

- Joint probability:
PMF Graphical Model

\[ P(L, R \mid X) \propto P(L)P(R)P(X \mid L, R) \]

- Graphically:

Maximum A Posteriori for Matrix Completion

\[
P(L, R|X) \propto P(L, R, X) = p(L)p(R)p(X \mid L, R) \\
\propto e^{-\frac{1}{2\sigma_u^2} \sum_{u=1}^n \sum_{i=1}^k L_{ui}^2} e^{-\frac{1}{2\sigma_v^2} \sum_{i=1}^m \sum_{i=1}^k R_{ui}^2} e^{-\frac{1}{2\sigma^2} \sum_{u,v} (L_{uv}R_{uv} - r_{uv})^2}\
\]
MAP versus Regularized Least-Squares for Matrix Completion

- MAP under Gaussian Model:
\[
\max_{L,R} \log P(L, R \mid X) = \max_{L,R} \log \frac{1}{Z} e^{-\frac{1}{2\sigma^2} \sum_u \sum_i L^2_{ui} - \frac{1}{2\sigma^2} \sum_v \sum_i R^2_{vi} - \frac{1}{2\sigma^2} \sum_{uv} (L_{uv} \cdot R_{uv} - r_{uv})^2 + \text{const}}
\]

- Least-squares matrix completion with L2 regularization:
\[
\min_{L,R} \frac{1}{2} \sum_{uv} (L_{uv} \cdot R_{uv} - r_{uv})^2 + \frac{\lambda_u}{2} ||L||^2_F + \frac{\lambda_v}{2} ||R||^2_F
\]

- Understanding as a probabilistic model is very useful! E.g.,
  - Change priors
  - Incorporate other sources of information or dependencies

What you need to know...

- Probabilistic model for collaborative filtering
  - Models, choice of priors
  - MAP equivalent to optimization for matrix completion
Case Study 4: Collaborative Filtering

Gibbs Sampling for Bayesian Inference

Posterior Computations

- MAP estimation focuses on point estimation:
  \[ \hat{\theta}^{MAP} = \arg \max_{\theta} p(\theta \mid x) \]

- What if we want a full characterization of the posterior?
  - Maintain a measure of uncertainty
  - Estimators other than posterior mode (different loss functions)
  - Predictive distributions for future observations

- Often no closed-form characterization (e.g., mixture models, PMF, etc.)
Bayesian PMF Example

- Latent user and movie factors:

- Observations
- Hyperparameters:

- Want to predict new movie rating:

Bayesian PMF vs. MAP PMF

\[
p(r_{uv}^* \mid X, \phi) = \int p(r_{uv}^* \mid L_u, R_v) p(L, R \mid X, \phi) dLdR
\]

- Relationship to MAP plug-in estimator:

©Emily Fox 2015
Monte Carlo methods:

Ideally:

Want posterior samples \((L^{(k)}, R^{(k)}) \sim p(L, R \mid X, \phi)\)

What can we sample from?

- Hint: Same reasoning as behind ALS, but sampling rather than maximization
Bayesian PMF Example

- For user u:
  \[ p(L_u \mid X, R, \phi_u) \propto p(L_u \mid \phi_u) \prod_{v \in V_u} p(r_{uv} \mid L_u, R_v, \phi_r) \]

- Symmetrically for R, conditioned on L (breaks down over movies)
- Luckily, we can use this to get our desired posterior samples

Gibb Sampling

- Want draws:

- Construct Markov chain whose steady state distribution is
- Then, asymptotically correct
- Simplest case:
Bayesian PMF Gibbs Sampler

- Outline of Bayesian PMF sampler

Bayesian PMF Results

- Netflix data with:
  - Training set = 100,480,507 ratings from 480,189 users on 17,770 movie titles
  - Validation set = 1,408,395 ratings.
  - Test set = 2,817,131 user/movie pairs with the ratings withheld.

Figure 2. Left panel: Performance of SVD, PMF, logistic PMF, and Bayesian PMF using 30D feature vectors, on the Netflix validation data. The y-axis displays RMSE (root mean squared error), and the x-axis shows the number of epochs, or passes, through the entire training set. Right panel: RMSE for the Bayesian PMF models on the validation set as a function of the number of samples generated. The two curves are for the models with 30D and 60D feature vectors.
Bayesian PMF Results

From Salakhutdinov and Mnih, ICML 2008

- Bayesian model better controls for overfitting by averaging over possible parameters (instead of committing to one)

![Table 1](image)

<table>
<thead>
<tr>
<th>D</th>
<th>Valid RMSE %</th>
<th>Test RMSE %</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>PMF</td>
<td>BPMF</td>
</tr>
<tr>
<td>40</td>
<td>0.9164</td>
<td>0.8994</td>
</tr>
<tr>
<td>40</td>
<td>0.9155</td>
<td>0.8968</td>
</tr>
<tr>
<td>60</td>
<td>0.9150</td>
<td>0.8954</td>
</tr>
<tr>
<td>150</td>
<td>0.9178</td>
<td>0.8931</td>
</tr>
<tr>
<td>300</td>
<td>0.9231</td>
<td>0.8920</td>
</tr>
</tbody>
</table>

Table 1. Performance of Bayesian PMF (BPMF) and linear PMF on Netflix validation and test sets.

The GraphLab Framework

Graph Based Data Representation

Update Functions User Computation

Scheduler

Consistency Model

©Emily Fox 2015
Update Functions
User-defined program: applied to \texttt{vertex} transforms data in \texttt{scope} of vertex

\begin{align*}
R[i] &= \alpha + (1 - \alpha) \\
&= \sum_{\text{scope i, neighbors}} \text{edge weight} \cdot \text{node weights} \\
&= \text{rank} \cdot \text{contained in scope of compute}
\end{align*}

Coordinate Descent for Matrix Factorization: Alternating Least-Squares

\begin{equation}
\min_{L, R} \sum_{(u, v): r_{uv} \neq 0} (L_u \cdot R_v - r_{uv})^2 + \lambda_u \| L \|_F + \lambda_v \| R \|_F
\end{equation}

- Fix movie factors, optimize for user factors
  - Independent least-squares over users
  \begin{equation}
  \min_{L_u} \sum_{v \in V_u} (L_u \cdot R_v - r_{uv})^2 + \lambda_u \| L \|_F
  \end{equation}

- Fix user factors, optimize for movie factors
  - Independent least-squares over movies
  \begin{equation}
  \min_{R_v} \sum_{u \in U_v} (L_u \cdot R_v - r_{uv})^2 + \lambda_v \| R \|_F
  \end{equation}

- System may be underdetermined:
  \begin{equation}
  \text{use regularization}
  \end{equation}

- Converges to \textbf{local optima}
Alternating Least Squares Update Function

- Update (in scope):
  - **Goal**: estimate \( L_u \)
  - **From scope**: gather factors and neighbors
  - Read all ratings across edges
  - Solve a local least squares problem

\[
\begin{align*}
\text{min}_{L_u} \sum_{v \in V_u} (L_u \cdot R_v - r_{uv})^2 & \quad \text{min}_{R_v} \sum_{u \in U_u} (L_u \cdot R_v - r_{uv})^2 \\
\end{align*}
\]

Bayesian PMF Gibbs Sampler

- **Outline of Bayesian PMF Gibbs Sampler**
  1. Initialize \( L^{(1)}, R^{(1)} \)
  2. For \( k = 1, \ldots, N_{iter} \)
     - (i) Sample hyperparams \( \phi^{(k)} \)
     - (ii) For each user \( u = 1, \ldots, n \) sample (in parallel)
       \[
       L_{u}^{(k+1)} \sim N(\tilde{\mu}_u, \tilde{\Sigma}_u)
       \]
     - (iii) For each move \( v = 1, \ldots, m \) sample (in parallel)
       \[
       R_{v}^{(k+1)} \sim N(\tilde{\mu}_v, \tilde{\Sigma}_v)
       \]
   where \( \tilde{\Sigma}_u^{-1} = \Sigma_u^{-1} + \sigma_r^{-2} \sum_{v \in V_u} R_v R_v^T \)
   \[
   \tilde{\mu}_u = \tilde{\Sigma}_u \left( \sigma_r^{-2} \sum_{v \in V_u} r_{uv} R_v + \Sigma_u^{-1} \mu_u \right)
   \]

©Emily Fox 2015
PMF Gibbs Sampling in GraphLab

\[ p(L_u \mid X, R, \phi_u) = N(\tilde{\mu}_u, \tilde{\Sigma}_u) \]
\[ \tilde{\Sigma}_u^{-1} = \Sigma_u^{-1} + \sigma_t^{-2} \sum_{v \in V_u} R_v R_v^T \]
\[ \tilde{\mu}_u = \tilde{\Sigma}_u \left( \sigma_t^{-2} \sum_{v \in V_u} r_{uv} R_v + \Sigma_u^{-1} \mu_u \right) \]

What you need to know...

• Idea of full posterior inference vs. MAP estimation
• Gibbs sampling as an MCMC approach
• Example of inference in Bayesian probabilistic matrix factorization model
• Implementation of vanilla sampler in GraphLab
Case Study 4: Collaborative Filtering

Matrix Factorization and Probabilistic LFM for Network Modeling

Network Data

- Structure of network data
Properties of Data Source

- Similarities to Netflix data:
  - Matrix
  - High-dimensional
  - Sparse

- Differences
  - Square
  - Binary

Matrix Factorization for Network Data

- Vanilla matrix factorization approach:

- What to return for link prediction?

- Slightly fancier:
Probabilistic Latent Space Models

- Assume features (covariates) of the user or relationship
- Each user has a “position” in a \(k\)-dimensional latent space

**Probability of link:**

\[
\log \text{odds } p(r_{uv} = 1 \mid L_u, L_v, x_{uv}, \beta) = \beta_0 + \beta^T x_{uv} - |L_u - L_v|
\]

\[
\log \text{odds } p(r_{uv} = 1 \mid L_u, L_v, x_{uv}, \beta) = \beta_0 + \beta^T x_{uv} + |L_u^T L_v|
\]

- Bayesian approach:
  - Place prior on user factors and regression coefficients
  - Place hyperprior on user factor hyperparameters
- Many other options and extensions (e.g., can use GMM for \(L_u \rightarrow\) clustering of users in the latent space)
What you need to know...

- Representation of network data as a matrix
  - Adjacency matrix
- Similarities and differences between adjacency matrices and general matrix-valued data
- Matrix factorization approaches for network data
  - Just use standard MF and threshold output
  - Introduce link functions to constrain predicted values
- Probabilistic latent space models
  - Model link probabilities using distance between latent factors