The Dinosaur Planet Approach to the Netflix Prize

David Lin* Lester Mackey** David Weiss***

*JP Morgan Stanley, **University of California, Berkeley,
***University of Pennsylvania

November 18, 2008

Outline

Outline

The Netflix Prize

- Netflix recommends movies to customers based on their preferences

The Netflix Prize

- Cinematch = Netflix movie recommender system
  - Collaborative filtering: patterns in the way users rate movies
  - Extract user tastes from past ratings
  - Predict other "Movies You'll ♡"
- Netflix Prize Challenge, Oct. 2, 2006
  - Beat Netflix recommender system, using Netflix data → win $1 million.

The Netflix Data

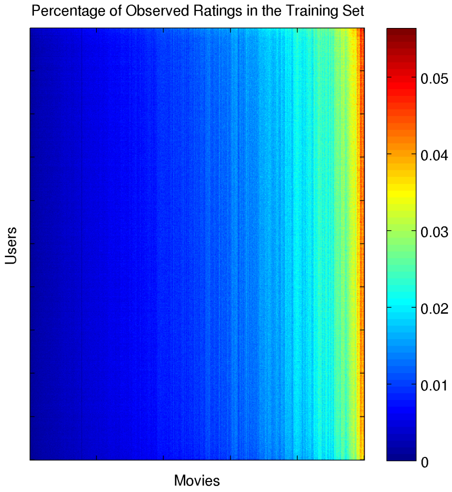
- Training set (TS)
  - 100 million examples (movie id, user id, date, rating)
  - 17,770 distinct movies
  - 480,189 distinct users
- Qualifying set (QS)
  - 2.8 million examples (movie id, user id, date)
  - Actual ratings withheld
  - Contains latest ratings of each user
    - Distribution fundamentally different from training set's!
- Probe set (PS)
  - 1.4 million examples (movie id, user id, date, rating)
  - Subset of training set
  - Same distribution as qualifying set

The Netflix Data

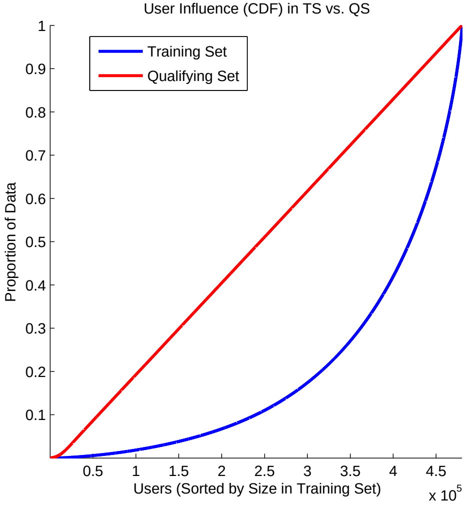
Key characteristics

- Largest publicly available dataset of its kind
- High sparsity
  - 17,770 x 480,189 $\approx$ 8.5 billion user-movie pairs
  - Only 1.18% of ratings are known
- No demographic data, just ratings
- Training and test sets have different distributions
  - Infrequent raters appear as often as frequent raters in QS

The Netflix Data



The Netflix Data



Evaluation Criteria

- Submit predictions of QS ratings to oracle (once per day)
- Score = root mean squared error (RMSE)
  - $\sqrt{\frac{1}{|QS|}\sum_{(u,m)\in QS}(pred_{(u,m)} - actual_{(u,m)})^2}$
- Cinematch QS RMSE: 0.9514
- 10% improvement (0.8563) $\Longrightarrow$ Grand Prize ($1 million)
- 1% improvement each year $\Longrightarrow$ Progress Prize ($50,000)
- Predicting error
  - Withhold probe set from training set
  - Use PS RMSE to predict QS RMSE
  - Cinematch PS RMSE: 0.9474

Team Dinosaur Planet

Brief Milestones

- **October, 2006:** Team Dinosaur Planet founded

Team Dinosaur Planet

Brief Milestones

- **October, 2006:** Team Dinosaur Planet founded
- **Spring, 2007:** DP enters “Top 10” on the leaderboard

Team Dinosaur Planet

Brief Milestones

- **October, 2006:** Team Dinosaur Planet founded
- **Spring, 2007:** DP enters "Top 10" on the leaderboard
- **Early September, 2007:** DP takes first place for 1 hour

Team Dinosaur Planet

Brief Milestones

- **October, 2006:** Team Dinosaur Planet founded
- **Spring, 2007:** DP enters “Top 10” on the leaderboard
- **Early September, 2007:** DP takes first place for 1 hour
- **Late September, 2007:** DP teams up with Team Gravity

Team Dinosaur Planet

Brief Milestones

- **October, 2006:** Team Dinosaur Planet founded
- **Spring, 2007:** DP enters “Top 10” on the leaderboard
- **Early September, 2007:** DP takes first place for 1 hour
- **Late September, 2007:** DP teams up with Team Gravity
- **October 1, 2007:** DP+Gravity retakes first place

Team Dinosaur Planet

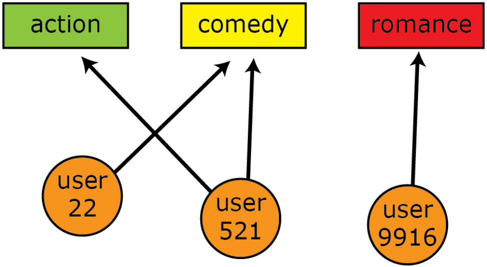
Brief Milestones

- **October, 2006:** Team Dinosaur Planet founded
- **Spring, 2007:** DP enters “Top 10” on the leaderboard
- **Early September, 2007:** DP takes first place for 1 hour
- **Late September, 2007:** DP teams up with Team Gravity
- **October 1, 2007:** DP+Gravity retakes first place
- **October 2, 2007:** 2nd place finish in Progress Prize

Outline

Clustering

- Divide users (or movies) into groups based on similarities



- Use group information to predict user ratings
  - e.g. The average action-lover gives Indiana Jones a 5
- **Hard clustering:** each user belongs to a single cluster
- **Soft or Fuzzy clustering:** each user fractionally belongs to all clusters

Clustering Models

General model

- U users, M movies, K clusters
- Represent user u as incomplete ratings vector $r_u \in \mathbb{R}^M$
  - e.g. $r_u = (1, 5, ?, ?, 3, ?, 4)$
- Represent each cluster k by a centroid vector $c_k \in \mathbb{R}^M$
  - Typically, c_k is average of user vectors in cluster k
- Minimize distance between users and their cluster centers

Hard clustering

- z_u := cluster of user u
- Minimize: $J(z) = \sum_{u=1}^{U} ||r_u - c_{z_u}||_2^2$

Fuzzy clustering

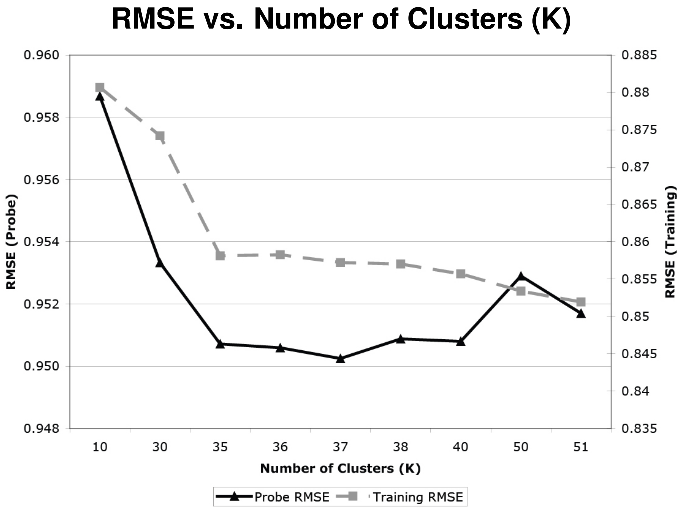
- $z_{u,k}$:= fractional belonging of u to cluster k, $\sum_{k=1}^{K} z_{u,k} = 1$
- Minimize: $J_\alpha(z) = \sum_{u=1}^{U} \sum_{k=1}^{K} z_{u,k}^{\alpha} ||r_u - c_k||_2^2$

Fuzzy C-Means

Fuzzy C-Means Algorithm (Dunn 1973, Bezdek 1981)

1. Choose number of clusters, K
2. Randomly assign users to clusters $\rightarrow z^{(0)}$
3. At each time step $t >= 0$, recompute
   - Cluster centers as weighted average of user vecs
     $$c_k^{(t)} = \frac{\sum_{u=1}^{U} z_{u,k}^{(t)\alpha} r_u}{\sum_{u=1}^{U} z_{u,k}^{(t)\alpha}}$$
   - User assignments based on distance to cluster centers
     $$z_{u,k}^{(t+1)} = \frac{1}{\sum_{j=1}^{K} \left(\frac{\left|\left| r_u - c_k^{(t)} \right|\right|_2}{\left|\left| r_u - c_j^{(t)} \right|\right|_2} \right)^{\frac{2}{\alpha - 1}}}$$
4. Repeat until assignments don't change (much)

Fuzzy C-Means Results



Best RMSE$_{\mathbf{Probe}}$: 0.9502 with 37 clusters

Fuzzy 3-way clustering

Motivation: Incorporate prior information

- Rating data naturally divide into “positive” {3,4,5} and “negative” {1,2} ratings

Algorithm

- Cluster on positive ratings {3,4,5} $\rightarrow \mathrm{E}\left[r_{u,m} | r_{u,m} \geq 3\right]$
- Cluster on negative ratings {1,2} $\rightarrow \mathrm{E}\left[r_{u,m} | r_{u,m} < 3\right]$
- Compute indicator vectors: $b_{u,m} = \mathbf{1}(r_{u,m} < 3)$
- Cluster on indicators $\rightarrow P(r_{u,m} < 3)$
- Predict
  - $\mathrm{E}\left[r_{u,m}\right] = P(r_{u,m} < 3) * \mathrm{E}\left[r_{u,m} | r_{u,m} < 3\right] + P(r_{u,m} \geq 3) * \mathrm{E}\left[r_{u,m} | r_{u,m} \geq 3\right]$

Best $\mathrm{RMSE}_{\mathrm{Probe}}$: 0.9499 with (8, 30, 12) clusters

Fuzzy 4-way clustering

Motivation: Confront weaknesses of 3-way clustering

- Positive vs. negative threshold is arbitrary
- Some 3-way clustering subproblems ignore subsets of the data

Algorithm

- For each $t \in \{2, 3, 4, 5\}$,
  1. Compute indicator vectors: $b_{u,m} = I(r_{u,m} < t)$
  2. Cluster on indicators $\rightarrow P(r_{u,m} < t)$
- Predict: $E(r_{u,m}) = 5 - \sum_{t=2}^{5} P(r_{u,m} < t)$

Best $\mathbf{RMSE_{Probe}}$: 0.9428 with (13, 12, 30, 35) clusters

Clustering on Errors

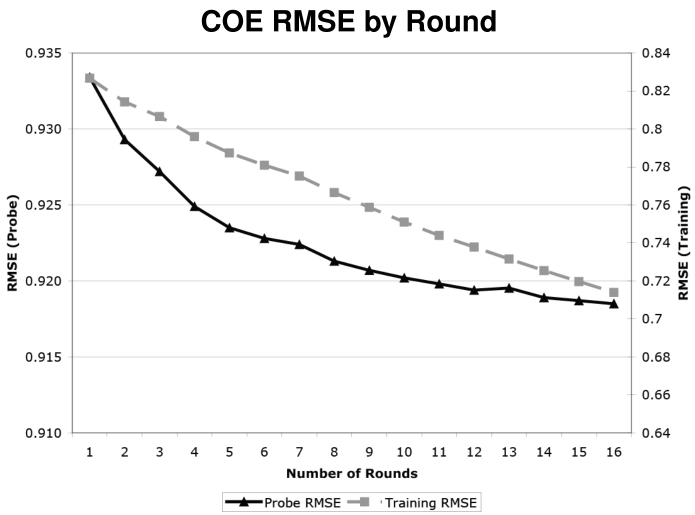
Motivation

- Cluster the residuals of clustering predictions
- Ensembles of clusters outperform single clustering

Algorithm

- Initialize $preds_u^0$ to predictions of any algorithm
- Initialize $s_u^0 = r_u$, the original ratings
- For $t = 1, ..., T$
  1. Update residuals $s_u^{t+1} = s_u^t - preds_u^t$
  2. Choose number of clusterings to perform, N_t
  3. For $c = 1, ..., N_t$
     Choose number of clusters $K_{t,c}$ in this clustering
     Cluster vectors s_u^t with $K_{t,c}$ clusters $\rightarrow preds_u^{t,c}$
  4. $preds_u^t = \frac{1}{N_t} \sum_{c=1}^{N_t} preds_u^{t,c}$

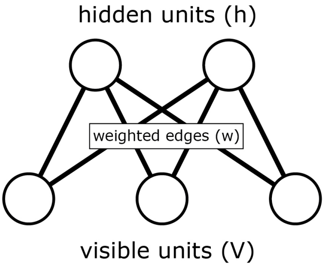
Clustering on Error Results



$\mathbf{RMSE_{Probe}}$: 0.9428 $\rightarrow$ 0.9187 after 16 rounds of COE

Restricted Boltzmann Machines - (RBMs)

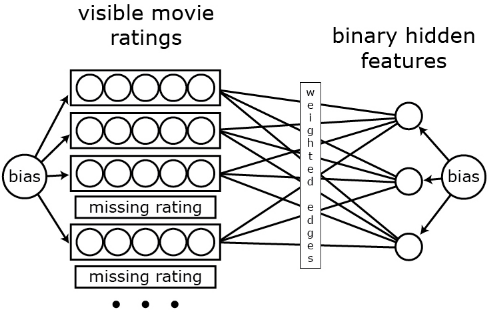
The Restricted Boltzmann Machine (Smolensky 1986)



- Bipartite, undirected graphical model
  - Visible layer, V: observed binary data
  - Hidden layer, H: latent binary "‘units"’
  - Weight parameters, W: interaction strength

RBM for Collaborative Filtering

RBM for CF Model (Salakhutdinov et al. 2007)



- Train separate RBM for each user
  - One visible "'softmax"' unit for each movie rated
    - Allow visible units to take on K (e.g. 5) values
  - Same number of hidden units across all RBMs
  - Weight matrix shared among all RBMs

RBM for Collaborative Filtering

RBM for CF Model (Salakhutdinov et al. 2007)

- User-specific Variables
  - V := binary matrix of user's ratings
    - $v_i^k = 1$ iff user gave rating k to ith movie
  - h := vector of binary hidden units
- Global Parameters
  - W := weights between visible and hidden units
  - b := hidden unit biases
  - c := visible unit biases
- Conditional distributions
  - $p(v_i^k = 1 | h, W, b, c) = \frac{\exp(c_i^k + \sum_{j=1}^F h_j W_{ij}^k)}{\sum_{l=1}^K \exp(c_i^l + \sum_{j=1}^F h_j W_{ij}^l)}$
  - $p(h_j = 1 | V, W, b, c) = \sigma(b_j + \sum_{i=1}^m \sum_{k=1}^K v_i^k W_{ij}^k)$

Learning in the RBM Model

Learning the parameters

- Goal: Choose parameters to maximize likelihood
- Potential Solution: Gradient ascent in log-likelihood
  - Problem 1: Analytical computation $\Rightarrow$ exponential time
  - Problem 2: Gibbs sampling $\Rightarrow$ high variance estimates
- Alternative: Gradient ascent in Contrastive Divergence

$$\Delta W_{ij}^k = \epsilon(< v_i^k h_j >_{data} - < v_i^k h_j >_T)$$
$$\Delta c_i^k = \epsilon(< v_i^k >_{data} - < v_i^k >_T)$$
$$\Delta b_j = \epsilon(< h_j >_{data} - < h_j >_T)$$

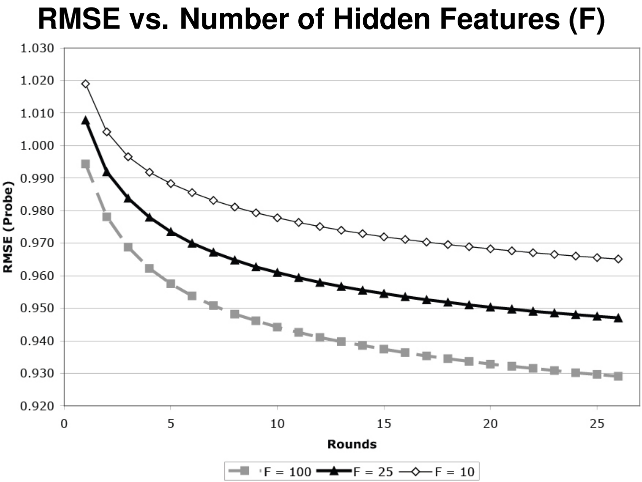
  - Compute $< . >_{data}$ terms analytically
  - Approximate $< . >_T$ terms with T rounds of Gibbs sampling

Prediction in the RBM Model

Making Predictions

- Mean field update
  $\hat{p}_j = p(h_j = 1 | V) = \sigma(b_j + \sum_{i=1}^{m} \sum_{k=1}^{K} v_i^k W_{ij}^k)$
- Predict expectation under conditional distribution
  $p(V | \hat{p}) = \frac{\exp(c_q^k + \sum_{j=1}^{F} \hat{p}_j W_{qj}^k)}{\sum_{l=1}^{K} \exp(c_q^l + \sum_{j=1}^{F} \hat{p}_j W_{qj}^l)}$

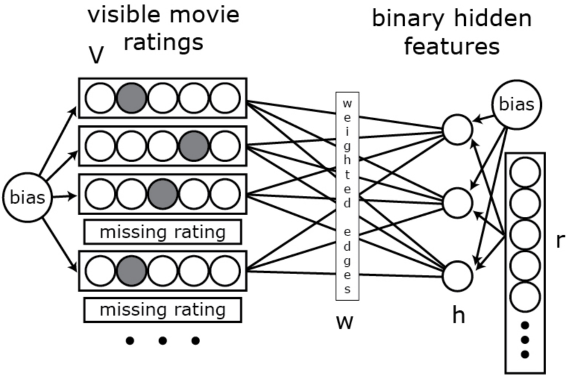
RBM Performance



Best $\mathbf{RMSE_{Probe}}$: 0.9104 with 200 hidden features

Conditional RBM

Conditional RBM for CF Model (Salakhutdinov et al. 2007)



Incorporate knowledge of who rated what (e.g. qualifying set)

Best RMSE$_{\text{Probe}}$: 0.9090 with 200 hidden features

Nearest Neighbor Methods

Intuition

- Predict $\hat{r}_{ui}$ based on user u's rating of "similar" movies to i

Details

- How to define similarity?
  - Inverse sqd. Euclidean distance: $\frac{1}{||r_m - r_n||^2}$
  - Cosine similarity: $\frac{\langle r_m, r_n \rangle}{||r_m|| \, ||r_n||}$
- How to weight neighbors?
  - Common approach: use similarities for weighted average:

$$\hat{r}_{ui} = \frac{\sum_{k \in \mathcal{S}^K_{ui}} s_{ik} r_{uk}}{\sum_{k \in \mathcal{S}^K_{uk}} s_{ik}}$$

  - Better approach: **fit** weights to optimize prediction accuracy

User-specific Least-squares KNN

Algorithm [Bell & Koren, 2007]

- Given a query for user u and movie i:
  1. Find the set S_{ui}^K of the K most similar movies to i that user u has rated.
  2. Solve for weights $\mathbf{w}$ that minimize the squared error of predictions for *other users* using S_{ui}^K as a basis:

$$\mathbf{w} = \operatorname*{argmin}_{\mathbf{w}} \sum_{v \neq u} \left(r_{vi} - \sum_{k \in \mathcal{S}_{ui}^K} w_k r_{vk} \right)^2$$

- What if other users have not rated each $j \in S_{ui}^K$?

KNN Approximation [Bell & Koren, 2007]

- x_{vj}: Rating of user v ($v \neq u$) on movie $j \in S_{ui}^K$
- y_v: Rating of user v on target movie i
- Optimal Solution:

$$\mathbf{w} = (\mathbf{X}^\top \mathbf{X})^{-1} \mathbf{X}^\top \mathbf{y},$$

- However, we can compute:

$$\mathbf{A} = \mathbf{X}^\top \mathbf{X}, \qquad A_{jk} \approx \frac{\sum_{v \in \mathcal{O}_{jk}} r_{vk} r_{vj}}{|\mathcal{O}_{jk}|}$$

$$\mathbf{b} = \mathbf{X}^\top \mathbf{y}, \qquad b_k \approx \frac{\sum_{v \in \mathcal{O}_{ik}} r_{vi} r_{vk}}{|\mathcal{O}_{ik}|}$$

- Approximate solution: $\mathbf{w} \approx \mathbf{A}^{-1} \mathbf{b}$

KNN Approximation

Implementation Details

- All possible elements of **A** can be precomputed in parallel $\longrightarrow$ prediction is fast
- Works well at postprocessing other algorithm's predictions
- **Best RMSE$_{\mathbf{Probe}}$**: 0.9184

Globally Optimized KNN

Motivation

- Fit the item-item similarity weights directly to maximize prediction accuracy
- Incorporate unlabelled data (i.e. viewed but not rated)

"Global KNN" Algorithm [Koren, 2008]

- $\mathcal{R}_u :=$ set of items rated by user u,
  $\mathcal{A}_u :=$ set of items *viewed* by user u (unlabelled instance)
- Predict weighted average of *all* data associated with user:

$$\hat{r}_{ui} = \mu + b_u + b_i + |\mathcal{R}_u|^{-\frac{1}{2}} \sum_{j \in \mathcal{R}_u} (r_{uj} - \beta_{uj}) w_{ij} + |\mathcal{A}_u|^{-\frac{1}{2}} \sum_{j \in \mathcal{A}_u} c_{ij}$$

Globally Optimized KNN

Implementation Details

- First estimate $\beta_{uj} = \mu + b_u + b_j$ using gradient descent
- Approximate $\mathcal{R}_u$ by a query-specific set $\mathcal{R}_{ui}^k = \mathcal{R}_u \cap S_i^k$
- Solve for $\mathbf{W}, \mathbf{C}$ using stochastic gradient descent to minimize $\sum(r_{ui} - \hat{r}_{ui})^2 + \lambda(||\mathbf{W}||^2 + ||\mathbf{C}||^2)$

Performance

- Koren [2008] reports better accuracy than user-specific kNN model when $K > 500$ and unlabelled data is used
- Preliminary $\mathbf{RMSE_{Probe}}$: 0.929 ($K = 300$, no $\mathbf{C}$)

"Super-Close" Neighbors

Motivation

- Some sets of items are extremely similar (e.g., T.V. show seasons, mini-series DVDs)
- Explicitly find such sets and correct for them

Finding "super-close" movies

- Simple correlation is not sufficient
- A large intersection size also not good enough
- Pairs must have a large intersection and small union

"Super-Close" Neighbors

Algorithm

- For movies i and j:
  - ρ_{ij} :=Pearson correlation between i and j
  - $i \Delta j$:= # of users who have seen i or j, but not both
  - Say i, j are "super-close" if:

$$d_{ij} = \frac{i \Delta j}{\min\{n_i, n_j\}} < d^\star, \rho_{ij} > \rho^\star$$

- Use heuristic to adjust $\hat{r}_{ui}$ closer to mean rating of any super-close movies

"Super-Close" Neighbors

Some *super-close* pairs

```
Corr: 94 - Sesame Street: Sing, Hoot & Howl - Sesame Street: Sing Along
Corr: 75 - Sesame Street: Sing, Hoot & Howl - Sesame Street: Cookie Monster's Best Bi
Corr: 88 - Sesame Street: Sing, Hoot & Howl - Sesame Street: Happy Healthy Monsters
Corr: 94 - Sesame Street: Sing, Hoot & Howl - Sesame Street: A Celebration of Me, Gro
Corr: 80 - Sesame Street: Sing, Hoot & Howl - Sesame Street: Fiesta!
Corr: 88 - Sesame Street: Sing, Hoot & Howl - Big Bird in Japan
Corr: 88 - Dr. Quinn, Medicine Woman: Season 5 - Dr. Quinn, Medicine Woman: Season 2
Corr: 93 - Dr. Quinn, Medicine Woman: Season 5 - Dr. Quinn, Medicine Woman: Season 4
Corr: 82 - I Love Lucy: Season 5 - I Love Lucy: Season 4
Corr: 85 - I Love Lucy: Season 5 - I Love Lucy: Season 3
Corr: 69 - I Love Lucy: Season 5 - The I Love Lucy 50th Anniversary Special
Corr: 71 - Le Petit Soldat - Les Carabiniers
Corr: 67 - The Simpsons: Season 1 - The Simpsons: Season 3
Corr: 72 - The Simpsons: Season 1 - The Simpsons: Season 2
Corr: 61 - The Simpsons: Season 1 - The Simpsons: Season 4
Corr: 85 - Frank Sinatra: The Main Event - Frank Sinatra: Ol' Blue Eyes Is Back
Corr: 64 - Poirot: The Murder of Roger Ackroyd - Miss Marple: Collection 2
Corr: 77 - Poirot: The Murder of Roger Ackroyd - Poirot: Peril at End House
Corr: 78 - Poirot: The Murder of Roger Ackroyd - Poirot: Dumb Witness
Corr: 66 - Poirot: The Murder of Roger Ackroyd - Poirot
Corr: 72 - Poirot: The Murder of Roger Ackroyd - Poirot: Hercule Poirot's Christmas
Corr: 68 - Poirot: The Murder of Roger Ackroyd - Miss Marple Mysteries: A Murder is A
Corr: 80 - Poirot: The Murder of Roger Ackroyd - Poirot: One Two Buckle My Shoe
Corr: 82 - Poirot: The Murder of Roger Ackroyd - Poirot: Lord Edgware Dies
Corr: 81 - Poirot: The Murder of Roger Ackroyd - Poirot: Murder on the Links
Corr: 72 - Poirot: The Murder of Roger Ackroyd - Poirot: Death in the Clouds
Corr: 76 - Poirot: The Murder of Roger Ackroyd - Poirot: The ABC Murders
Corr: 77 - Poirot: The Murder of Roger Ackroyd - Poirot: The Mysterious Affair at Sty
Corr: 75 - Poirot: The Murder of Roger Ackroyd - Poirot: Murder in Mesopotamia
Corr: 84 - Poirot: The Murder of Roger Ackroyd - Poirot: Hickory Dickory Dock
Corr: 93 - Thomas & Friends: Thomas's Snowy Surprise - Thomas & Friends: Hooray for T
Corr: 87 - Thomas & Friends: Thomas's Snowy Surprise - Thomas & Friends: The Early Ye
Corr: 85 - Thomas & Friends: Thomas's Snowy Surprise - Thomas & Friends: It's Great t
```

Matrix Factorization

Intuition

- Ratings are the sum of interactions between user tastes and movie properties
- Tastes/properties quantified as vectors:

$$\hat{r}_{ui} = \sum_k p_{uk} q_{ik} = \mathbf{p}_u \mathbf{q}_i^\top$$

Model

- Ratings data is a sparse $N \times M$ matrix:

$$\mathbf{R} = \begin{bmatrix} ? & ? & 1 & \dots & 4 \\ 3 & ? & ? & \dots & ? \\ ? & 5 & ? & \dots & 5 \end{bmatrix}$$

- Factorize $\mathbf{R}$ as the product of two rank K matrices:

$$\mathbf{R} \approx \mathbf{P}\mathbf{Q}^\top$$

Learning the MF Model

Minimizing Reconstruction Error

- Many standard MF algorithms minimize squared reconstruction error,

$$\operatorname*{argmin}_{\mathbf{P},\mathbf{Q}} ||\mathbf{R} - \mathbf{P}\mathbf{Q}^\top||^2$$

- E.g., SVD/PCA, NNMF
- We are only interested in constructing the qualifying set: only 0.03% of **R**!

Learning the MF Model

Practical Solutions

- Minimize regularized squared error on TS examples:

$$L = \sum_{u,i \in \mathcal{T}} \left(r_{ui} - \mathbf{p}_u \mathbf{q}_i^\top \right)^2 + \lambda_p \sum_u ||p_u||^2 + \lambda_q \sum_i ||q_i||^2$$

- Fit parameters via cross-validation
- Use algorithms that operate per-example (stochastic gradient descent) or per-user/movie (alternating least squares)
- Blithely ignore the problem of local minima and convergence testing

Gradient Descent

Algorithm:

- For each record r_{ui} in the training set:
  1. Calculate residual error: $e \leftarrow r_{ui} - \mathbf{p}_u \mathbf{q}_i^\top$
  2. Update user factors: $p_{uk} \leftarrow p_{uk} + \eta e(q_{ik} - \lambda_p p_{uk})$
  3. Update movie factors: $q_{ik} \leftarrow q_{ik} + \eta e(p_{uk} - \lambda_q q_{ik})$
- η is learning rate
- Stop after T iterations

In practice:

- Easy to implement, fast
- Improvements: early stopping via validation set, learning rate decay, etc.

Best RMSE$_{\text{Probe}}$: 0.9101

Alternating Least Squares

Algorithm

- Fully observed case: given $\mathbf{R}$, initial $\mathbf{P}, \mathbf{Q}$:
  1. Update $\mathbf{P} \leftarrow \mathbf{R}\mathbf{Q}(\mathbf{Q}^\top\mathbf{Q} + \lambda_p\mathbf{I})^{-1}$
  2. Update $\mathbf{Q} \leftarrow (\mathbf{P}^\top\mathbf{P} + \lambda_q\mathbf{I})^{-1}\mathbf{P}^\top\mathbf{R}$
  3. Stop after T iterations

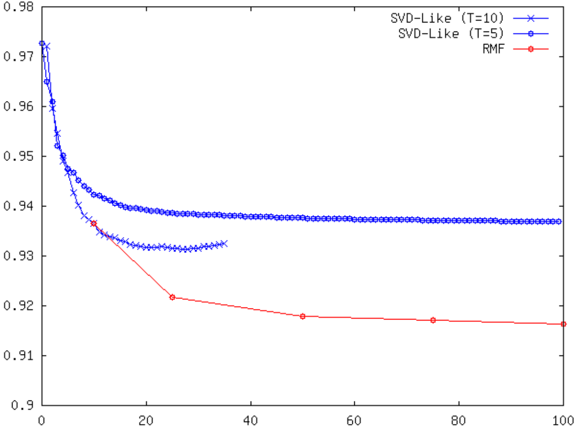
How to efficiently account for missing values?

- For each factor $\mathbf{p}_u$:
  1. $\mathbf{r}_{(u)} \equiv$ movie ratings of user u
  2. $\mathbf{Q}_{(u)} \equiv$ movie factors for rated movies
  3. Update $\mathbf{p}_u \leftarrow \mathbf{r}_{(u)}\mathbf{Q}_{(u)}(\mathbf{Q}_{(u)}^\top\mathbf{Q}_{(u)} + \lambda_p\mathbf{I})^{-1}$
- Use similar strategy to update each $\mathbf{q}_i$

Implementation Notes

- **SVD-like implementations**
  - Basic procedure:
    1. Store single array of dataset residuals
    2. Learn MF model with $K = 1$
    3. Update residuals array, discard factors
    4. Repeat
  - Decrease T with each factor added to prevent overfitting
- **Non-negativity constraints (NMF, semi-NMF, etc.)**
  - Computing exact solutions is slow
  - After each ALS/Gradient Descent update, *rectify* **P** and **Q** $(x : x < 0 \leftarrow 0)$
  - Will induce sparsity!
- **More Approaches: Gibbs Sampling, 0-Imputation**

Sample Learning Curves



MF for Collaborative Filtering

Intuition - Can we avoid parameterizing users?

- In real-world setting, users drop in/out of database all the time – unrealistic for Netflix to store/update $\mathbf{P}$
- Observe: given arbitrary $\mathbf{Q}_{(u)}$, can solve for p_u
- Goal: build a MF model that never stores an explicit representation for each user

MF for Collaborative Filtering

"Asymmetric" Factorization

- Library shelf idea: a user's tastes are the sum of those tastes *indicated by* movies in their library
- Any two users who have viewed exactly the same set of movies are the same

$$\mathbf{p}_u = \sum_{j \in \mathcal{A}_u} \mathbf{y}_j = \sum_j A_{uj} \mathbf{y}_j \quad \rightarrow \quad \hat{r}_{ui} = \mathbf{q}_i^\top \sum_{j \in \mathcal{A}_u} \mathbf{y}_j,$$

$$\mathbf{P} = \mathbf{AY} \quad \rightarrow \quad \mathbf{R} \approx \mathbf{AYQ}^\top$$

- $\mathbf{A}$ is binary indicator matrix (not used in practice!)
- $\mathbf{Y}, \mathbf{Q}$ are both $M \times K$ "movie factors"

Best RMSE$_{\textbf{Probe}}$: 0.94133

Asymmetric Factor Models

“NSVD” - Paterek (2007)

$$\hat{r}_{ui} = \left(|\mathcal{A}_u|^{-\frac{1}{2}} \sum_{j \in \mathcal{A}_u} \mathbf{y}_j \right) q_i^\top$$

“SVD++” - Koren (2008)

$$\hat{r}_{ui} = \left(\mathbf{p}_u + |\mathcal{A}_u|^{-\frac{1}{2}} \sum_{j \in \mathcal{A}_u} \mathbf{y}_j \right) \mathbf{q}_i^\top$$

Takacs et al. (2008)

$$\hat{r}_{ui} = \mathbf{p}_u \mathbf{q}'^\top_i + |\mathcal{A}_u|^{-\frac{1}{2}} \sum_{j \in \mathcal{A}_u} \mathbf{y}_j \mathbf{q}_i^\top$$

Co-Training

Intuition

- If two different algorithms are both correct, they should agree on unlabeled data
- "Co-Training" algorithm first proposed by Blum & Mitchell (1998): Enforce agreement on unlabeled data during the training process
- A can help compensate for mistakes by B, vis versa

Integrated Models

- Training global KNN model and "SVD++" simultaneously results in most accurate single model to date [Koren, 2008]
- Can we do even better by regularizing to enforce agreement on Qualifying Set?

Co-Regularization Experiment

Co-Regularized KNN-MF

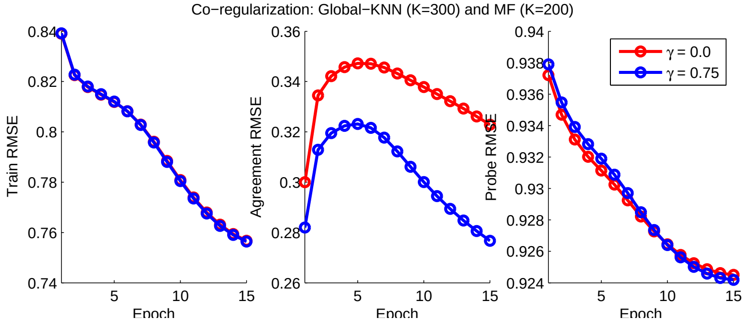
- Simplified integrated model (no implicit data):

$$\hat{r}_{ui} = \mu + b_u + b_i + |\mathcal{R}_u|^{-\frac{1}{2}} \sum_{j \in \mathcal{R}_u} (r_{uj} - \beta_{uj}) w_{ij} + \mathbf{p}_u \mathbf{q}_i^\top$$

- Minimize co-regularized objective (gradient descent):

$$\begin{aligned} L &= \sum_{u,i \in \mathcal{TS}} (\hat{r}_{ui} - r_{ui})^2 \\ &+ \lambda \left[\mu + b_u + b_i + \sum_{i,j} w_{ij}^2 + \sum_u ||\mathbf{p}_u||^+ \sum_i ||\mathbf{q}_i||^2 \right] \\ &+ \gamma \left[\sum_{u,i \in \mathcal{QS}} \left(\mathbf{p}_u \mathbf{q}_i^\top - |\mathcal{R}_u|^{-\frac{1}{2}} \sum_{j \in \mathcal{R}_u} (r_{uj} - \beta_{uj}) w_{ij} \right)^2 \right] \end{aligned}$$

Co-Regularization Experiment



- $\gamma = 0.00 \rightarrow$ **RMSE$_{\text{Quiz}}$** $=$0.9258
- $\gamma = 0.75 \rightarrow$ **RMSE$_{\text{Quiz}}$** $=$0.9254

Outline

Model Blending

Why combine models?

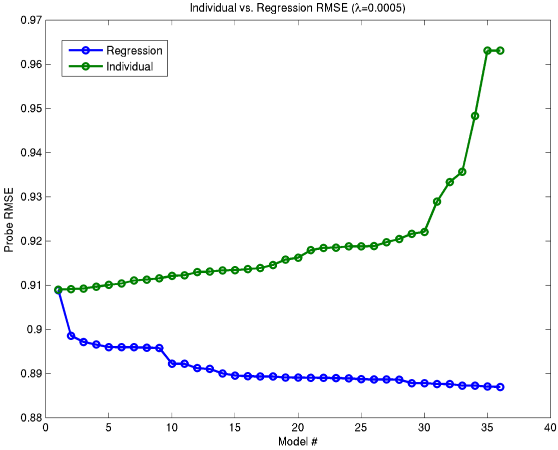
- Diminishing returns from optimizing a single algorithm
- Different models capture different aspects of the data
- Statistical motivation
  - If X_1, X_2 uncorrelated with equal mean, $Var(\frac{X_1}{2} + \frac{X_2}{2}) = \frac{1}{4}(Var(X_1) + Var(X_2))$
  - Moral: Errors of different algorithms can cancel out

Model Blending

Probe set Ridge Regression

- Linearly combine algorithm predictions
- Let columns of **P** = PS predictions of each algorithm
- Let **y** = true PS ratings
- Solve for (near) optimal blending coefficients, β
  $\min_\beta ||\mathbf{y} - \mathbf{P}\beta||^2 + \lambda\,||\beta||^2$
- Solution: $\beta = (\mathbf{P}^\top\mathbf{P} + \lambda\mathbf{I})^{-1}\mathbf{P}^\top\mathbf{y}$
- λ = ridge/regularization parameter
  - Reduces overfitting
  - Guarantees invertibility

Blending Demonstration



"No-train" Regressors

The search for anything that might help explain the ratings in a different way:

"No-train" Regressors

The search for anything that might help explain the ratings in a different way:

- user size

"No-train" Regressors

The search for anything that might help explain the ratings in a different way:

- user size
- date

"No-train" Regressors

The search for anything that might help explain the ratings in a different way:

- user size
- date
- 1/(user size+1)

"No-train" Regressors

The search for anything that might help explain the ratings in a different way:

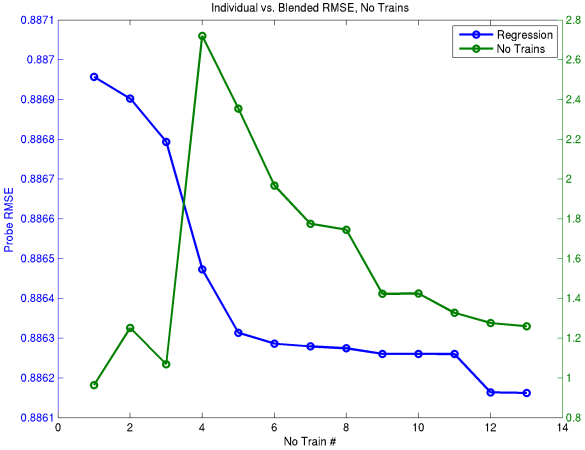
- user size
- date
- 1/(user size+1)
- average inverse size of all users that saw the movie

"No-train" Regressors

The search for anything that might help explain the ratings in a different way:

- user size
- date
- 1/(user size+1)
- average inverse size of all users that saw the movie
- log(1+number of 2 ratings this user has given)

No trains in practice



Quantifying Interactions

- **Explicitly create new regressors out of interactions between existing ones**
  - Polynomial (e.g., $z_n = x_{n,i} x_{n,j}$)
  - Functions (e.g., $z_n = \log(x_n)$)
  - Example: *Interacting #1 with all:* 0.8870 $\rightarrow$ 0.8864
  - Example: *Interacting #1, #31 with all:* 0.8870 $\rightarrow$ 0.8861
- **Downsides**
  - Overfitting
  - Dramatically increases runtime!
    Matrix inversion with all M^2 interactions = $O(M^6)$
- **Must make interaction-adding feasible**

Greedy Interaction Selection

Algorithm:

- Precompute $\mathbf{X}^\top\mathbf{X}$, PS blended RMSE r, and $k = M + 1$
- For $i, j \in \{1, \ldots, M\}$:
  1. Compute interaction $\mathbf{z} = \mathbf{x}_i \cdot \mathbf{x}_j$
  2. Compute k'th row/column of $\mathbf{X}^\top\mathbf{X} = \mathbf{z}^\top\mathbf{X}$
  3. Compute new regression and record Probe Set RMSE r'
  4. If $r' < r - \epsilon$, increment k and set $r \leftarrow r'$

In practice

- Adjusting ϵ adjusts the number of accepted interactions
- Still too slow! (36 hours)

Collapsed Interactions

Modified Algorithm:

- For $i \in \{1, \ldots, M\}$:
  1. Perform regression on initial regressors and interactions $\{(i,i),(i,i+1),\ldots,(i,M)\}$
  2. Compute $\mathbf{y}_i$, the optimal blend using regression solution
  3. Replace original regressor $\mathbf{x}_i$ with new regressor $\mathbf{y}_i$

Benefits

- Feasible: regression with at most $2M$ predictors
- Probe Set: $0.8841 \rightarrow 0.8809$
- Qual Set: $0.8777 \rightarrow 0.8757$

Bagged Interactions

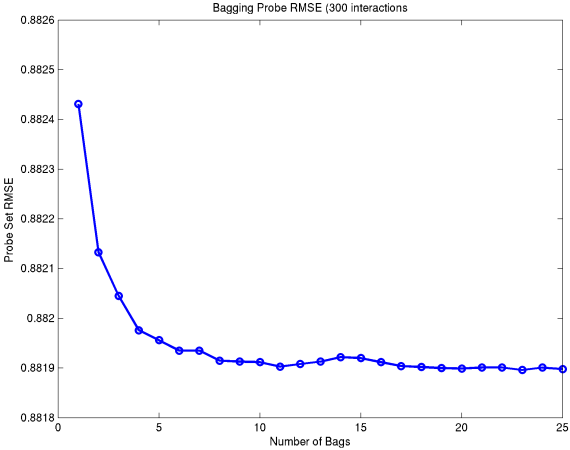
Algorithm:

- For $s \in \{1, \ldots, S\}$:
  1. Generate bootstrap replicate $\mathbf{X}^{(s)}, \mathbf{y}^{(s)}$
  2. Add K random interaction terms to $\mathbf{X}^{(s)}$
  3. Solve for $\beta^{(s)}$ using standard ridge procedure
- $\beta^{final} = \frac{1}{S} \sum_s \beta^{(s)}$

Benefits

- Easy to control run-time complexity
- Bagging helps reduce overfitting (general principle)

Bagged Interactions



Outline

Summary of Results

Algorithm Class	Probe	Qual
Clustering	[0.9187-0.9502]	N/A
KNN	0.9184	N/A
MF	[0.9101 - 0.9289]	N/A
RBM	[0.9090 - 0.9104]	N/A
Ensemble of 50 predictors	0.8861	N/A
Ensemble + Correlation	0.8854	N/A
Ensemble + No Trains + Corr	0.8841	0.8777
All + Greedy Interactions	0.8806	0.8756
All + Collapsed Interations	0.8809	0.8757
All + Bagging Interactions	0.8813	0.8753
DP Best + Gravity Best	0.8702	0.8675

References

- R.M. Bell and Y Koren (2007), “Scalable Collaborative Filtering with Jointly Derived Neighborhood Interpolation Weights”, *Proc. IEEE International Conference on Data Mining (ICDM ’07)*
- J. C. Bezdek (1981): “Pattern Recognition with Fuzzy Objective Function Algoritms”, *Plenum Press*, New York
- J. C. Dunn (1973): “Fuzzy Relative of the ISODATA Process and Its Use in Detecting Compact Well-Separated Clusters” *Journal of Cybernetics 3*: 32-57
- R. Salakhutdinov, A. Mnih, and G. Hinton. Restricted Boltzmann Machines for collaborative filtering. *Proceedings of the 24th International Conference on Machine Learning*, 2007.
- P. Smolensky. Information processing in dynamical systems: foundations of harmony theory. In D.E. Rumehart and J.L. McClelland, editors, *Parallel Distributed Processing: Explorations in the Microstructure of Cognition. Volume 1: Foundations.* McGraw-Hill, New York, 1986.