Algorithms for Collaborative Filtering

or

"How to Get Half Way to Winning $1 million from Netflix"

Todd Lipcon
Advisor: Prof. Philip Klein

Brown University Department of Computer Science

The Real-World Problem

- E-commerce sites would like to make personalized product recommendations to customers
- Often, there is no data about the customers, and/or the products are hard to describe explicitly (e.g. movies, music, art, books)

The Simplified Problem

- Regression/prediction problem
- Given a {user, item} pair we would like to predict the user's ordinal rating of that item
- In the case of the Netflix Prize, we are scored based on squared error of our predictions

Collaborative Filtering

- We *do* have past preference data for repeat users
  - Purchase history (binary)
  - Ratings (ordinal)
- We can *infer* traits of users and items from this preference data

The Problem Take 2

(this time with equations)

- We form a *Rating Matrix* $A \in \Re^{m \times n}$ *where* A_{ij} is the rating by user i on item j
- In Netflix, 99% of the entries in A are missing
- We would like to solve:

$$X = \underset{X \in \Re^{m \times n}}{\operatorname{argmin}} \sum_{ij \in A} (A_{ij} - X_{ij})^2$$

subject to some constraint or regularization penalty on X to ensure generalization to new examples ij

Item-KNN Approach

- We assume that items in the system can be described by their similarity to other items
- We can then predict that a given user will assign similar ratings to similar items
- We build a list of neighbors for each movie based on Pearson Correlation between their overlapping rating data
- We adjust correlation by calculating a confidence interval

Item-KNN Prediction

	RMSE	NMAE
Netflix	0.9411	0.4274
MovieLens	0.9126	0.4208

- State of the art ("Ensemble MMMF") achieves NMAE of 0.4054 on MovieLens
- Best Netflix result is RMSE = 0.8831
- CineMatch RMSE = 0.9514

Item-KNN Data Mining

Movies Similar to *Finding Nemo (Fullscreen)*

Correlation	Movie Title
0.650	*Finding Nemo (Widescreen)*
0.316	*Monsters, Inc.*
0.281	*A Bug's Life*
0.259	*Toy Story*
0.256	*Toy Story 2*
0.251	*The Incredibles*
0.243	*Shrek 2*
0.226	*Ice Age*
0.218	*The Lion King: Special Edition*
0.215	*Harry Potter and the Prisoner of Azkaban*

Same movie!

Pixar

CG/Kids

Animation

Kids

Matrix Factorization

We replace objective

$$X = \underset{X \in \Re^{m \times n}}{\operatorname{argmin}} \sum_{ij \in A} \left(A_{ij} - X_{ij}\right)^2$$

with the factorized objective

$$X = \underset{U \in \Re^{m \times r}, V \in \Re^{n \times r}}{\operatorname{argmin}} \sum_{ij \in A} \left(A_{ij} - (UV^T)_{ij}\right)$$

which is easier to learn due to smaller number of free parameters $m + n$ instead of $m \cdot n$

Learning U and V

- The factorized objective isn't convex
- We'd rather just learn the matrices one column at a time such that each additional column reduces the error as much as possible
- Learning the columns u and v is still non-convex, but works well in practice anyway
- Learning the columns incrementally is an example of a gradient boosting machine [Friedman]

Gradient Boosting Matrix Factorization

- Pseudocode:

$$\forall ij \in X : X_{ij} = \textsc{Baseline}(i, j)$$

for i = 1 to M

$$\forall ij \in X : \tilde{X}_{ij} = -\frac{\partial}{\partial X_{ij}}(A_{ij} - X_{ij})^2$$

any convex loss function can be substituted here!

$$\{u, v\} = \textsc{UVLearn}(\tilde{X})$$

shrinkage parameter (more later)

$$U = [U\ \nu u]; V = [V\ \nu v]; X = X + \nu u v^T$$

end for

UVLearn Objective

- Aim to learn best rank-1 approximation of the input matrix $\tilde{X}$
- Given $X \in \Re^{m \times n}$, solve:

$$\{u, v\} = \underset{u \in \Re^m, v \in \Re^n}{\operatorname{argmin}} \sum_{ij} (X_{ij} - u_i v_j)^2$$

- But this will overfit training data – needs regularization!

UVLearn Objective

(regularized)

- Given $X \in \Re^{m \times n}$ and regularization parameter λ, solve:

$$\{u, v\} = \underset{u \in \Re^m, v \in \Re^n}{\operatorname{argmin}} \left[\sum_{ij} (X_{ij} - u_i v_j)^2 + \lambda \sum_{i=1}^{m} |u_i - \bar{u}|^2 + \lambda \sum_{j=1}^{n} |v_j - \bar{v}|^2 \right]$$

UVLearn Regularization

- Penalization is based around distance from "empirical prior" means $\bar{u}$ and $\bar{v}$
- We simultaneously optimize the parameters u and v along with their changing means

UVLearn Optimization

- Stochastic Gradient Descent – partial derivatives:

$$\frac{\partial E_{ij}}{\partial u_i} = 2v_j(u_i v_j - R_{ij}) + 2\lambda(u_i - \bar{u})$$

$$\frac{\partial E_{ij}}{\partial v_j} = 2u_i(u_i v_j - R_{ij}) + 2\lambda(v_j - \bar{v})$$

$$\frac{\partial E_{ij}}{\partial \bar{u}} = -2\lambda(u_i - \bar{u})$$

$$\frac{\partial E_{ij}}{\partial \bar{v}} = -2\lambda(v_j - \bar{v})$$

UVLearn Optimization

- Stochastic GD update equations:

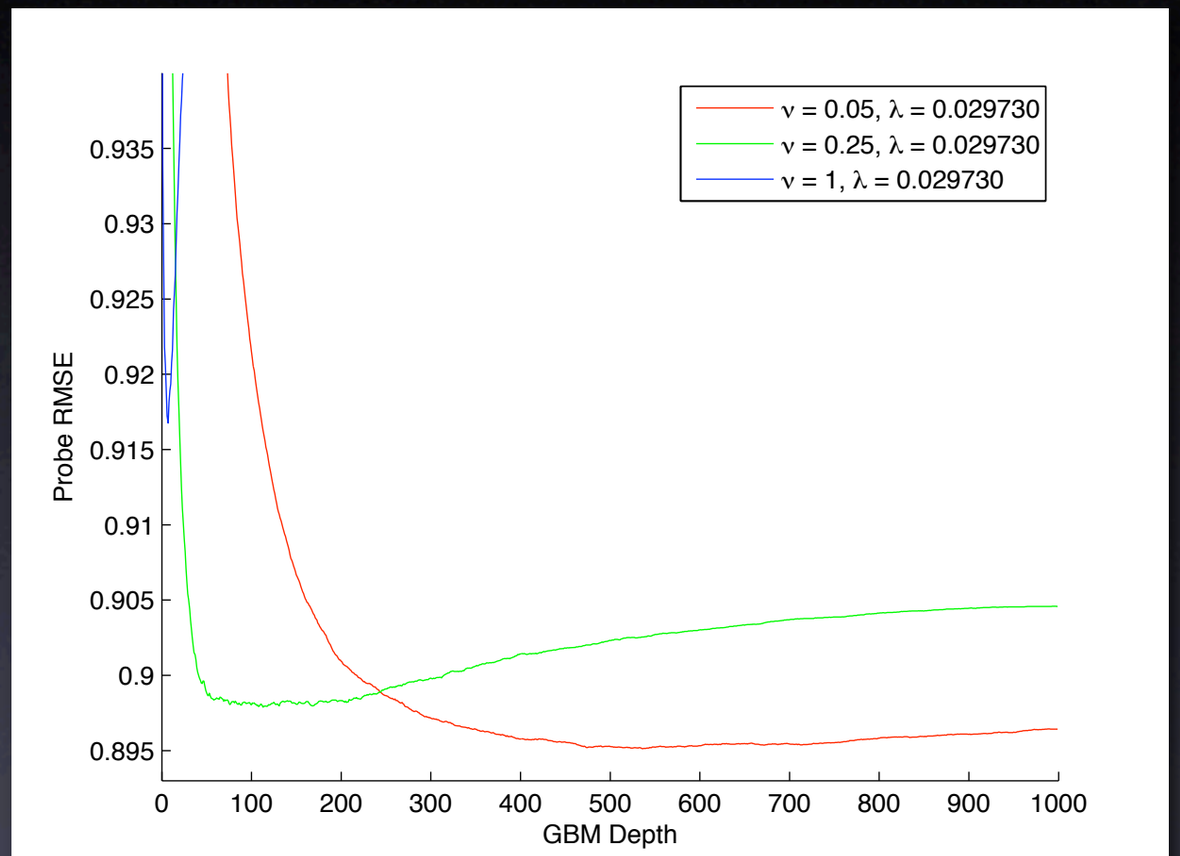
$$u_i' = u_i - \eta\left(2v_j(u_i v_j - X_{ij}) + 2\lambda(u_i - \bar{u})\right)$$
$$v_j' = v_j - \eta\left(2u_i(u_i v_j - X_{ij}) + 2\lambda(v_j - \bar{v})\right)$$
$$\bar{u}' = \bar{u} - \eta\left(-2\lambda(u_i - \bar{u})\right)$$
$$\bar{v}' = \bar{v} - \eta\left(-2\lambda(v_j - \bar{v})\right)$$

$$H\left(E(u_i, v_j, \bar{u}, \bar{v})\right) =$$
$$\begin{bmatrix} 2v_j^2 + 2\lambda & 4u_i v_j - 2R_{ij} & -2\lambda & 0 \\ 4u_i v_j - 2R_{ij} & 2u_i^2 + 2\lambda & 0 & -2\lambda \\ -2\lambda & 0 & 2\lambda & 0 \\ 0 & -2\lambda & 0 & 2\lambda \end{bmatrix}$$

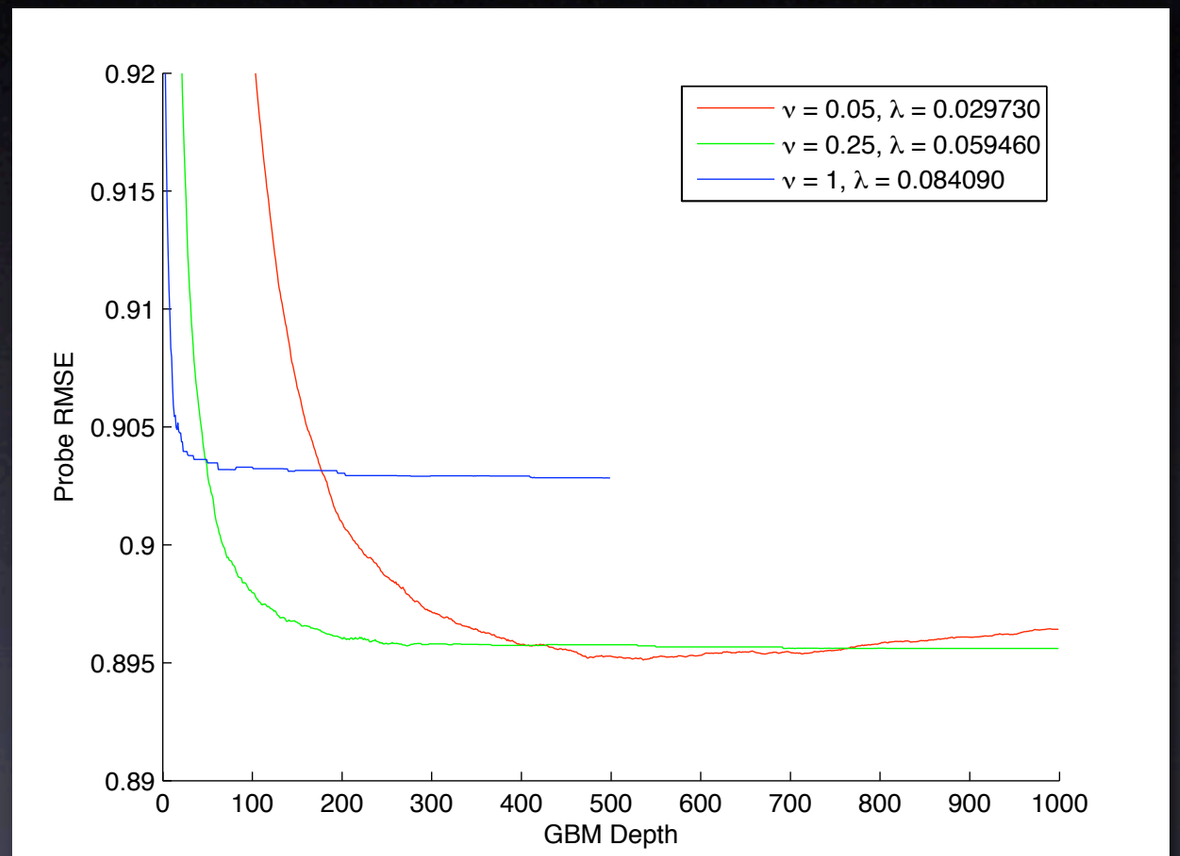
GBMF Regularization

- UVLearn regularization parameter λ
- UVLearn GD stopping threshold τ
- Gradient boosting shrinkage parameter ν
- Number of gradient boosting iterations M

Shrinkage Effects



Different shrinkage, same UVLearn regularization



Different shrinkage, best UVLearn regularization

GBMF Prediction

	RMSE	NMAE
Netflix	0.9134	0.4099
MovieLens	0.8859	0.4051

- State of the art ("Ensemble MMMF") achieves NMAE of 0.4054 on MovieLens
- Best Netflix result is RMSE = 0.8831
- CineMatch RMSE = 0.9514

Bagged GBMF

- Bootstrap aggregation (*bagging*) - general variance-reduction framework that does not increase bias [Breiman]
- We construct ensembles of 60 GBMF models trained on bootstrap samples of the training set
- Ensemble predictions are the mean of member predictions

Bagged GBMF Prediction

	RMSE	NMAE
Netflix	??	??
MovieLens	0.8850	0.4045

Netflix jobs are still running and getting better

Weren't able to do good grid search yet even for MovieLens because of resource limitations on SGE cluster

- State of the art ("Ensemble MMMF") achieves NMAE of 0.4054 on MovieLens
- Best Netflix result is RMSE = 0.8831
- CineMatch RMSE = 0.9514

Combining Predictors

- A single data set has many different types of component data (varying sparsity, etc)
- A single algorithm (or choice of tuning parameters for an algorithm) may not give the best results for every type of component data
- We can do better by smartly blending between different predictors

Learning Combinations

- Train individual predictors on most of the data
- Train combination algorithm on remaining data using *k*-fold cross-validation for model selection
- Then retrain individual predictors on the full data and use the combination predictor learned above

Learning Combinations

- For each entry in the validation set, we create a vector with movie mean, user mean, movie count, user count, and the predictions by all component predictors.
- We then can use standard regression techniques to learn the combination function.

Linear Combination

- Use simple least squares regression:

	RMSE	NMAE
Netflix	0.9092	0.4061
MovieLens	0.8817	0.4062

- Add 1st-order interaction terms:

	RMSE	NMAE
Netflix	0.9062	0.4049
MovieLens	0.8758	0.4043

- Adding 2nd-order interaction terms overfits

Support Vector Machine Combination

- Use nu-SVR support vector regression to minimize squared loss
- Use support vector ordinal regression to minimize absolute loss
- Work still in progress - initial results for NMAE significantly improve on Ensemble MMMF (MovieLens NMAE ~ 0.4000)

Other Strong Netflix Approaches

- Restricted Boltzmann Machines (ML @ UToronto team) [Salakhutdinov, Mnih, and Hinton 2007]
- Probabilistic Latent Semantic Analysis ("PLSA" team) [Hofmann 2003]
- Gibbs Sampling of Latent Variables (Paul Harrison)
- Most of the top teams are doing combination of some kind

Future Work

- Better grid search for bagging experiments
- Stochastic Meta-Descent optimization
- Continued work on SVR combination
- "Conditional" rating prediction
- Alternative loss functions (GBMF optimizes *any* convex loss)

Questions?