The BigChaos Solution to the Netflix Prize

Presented by:
Chinfeng Wu

Outline

- The Netflix Prize
- The team "BigChaos"
- Algorithms
- Details in selected algorithms
- End-Game
- Conclusion
- Q & A

The Netflix Prize

- Participants download training data to derive their algorithm
- Submit predictions for 3 million ratings in "Held-Out Data" (could submit multiple times, limit of once/day)
- Prize
    - $1 million dollars if error is 10% lower than Netflix current system
    - Annual progress prize of $50,000 to leading team each year

More on Netflix

- Training Data:
  - 100 million anonymized ratings (matrix is 99% sparse), generated by 480k users x 17.7k movies between Oct 1998 and Dec 2005
  - Rating = [user, movie-id, time-stamp, rating value]
  - Users randomly chosen among set with at least 20 ratings
- Held-Out Data:
  - 3 million ratings- True ratings are known only to Netflix
  - 1.5m ratings are quiz set, scores posted on leaderboard
  - The rest 1.5m ratings are test set, scores known only to Netflix to determining final winner

Scoring of Netflix

- Use RMSE (Root Mean Squared Error)
- RMSE Baseline Scores on Test Data
  - 1.054 -just predict the mean user rating for each movie
  - 0.953 -Netflix's own system (Cinematch) as of 2006
  - 0.941 -nearest-neighbor method using correlation
  - 0.857 -required 10% reduction to win $1 million

The Team “BigChaos”

- Team Member: Michael Jahrer & Andreas Toscher, 2 master students from Austria
- Collaborate with the team “BellKor” to win Netflix Progress Prize 2008
- Collaborate with the teams “BellKor”, “Pragmatic Theory” to win Netflix Grand Prize

Algorithms

- Automatic Parameter Tuner:
  - APT1 - A simple random search method, used to find parameters lead to local minimum RMSE.
  - APT2 - A structured coordinate search, used to minimize the error function.
- Basic Predictors: Use mean rating for each movie.

Algorithms (continue)

- Weekday Model (WDM): Predict ratings on the basis of weekday means. Calculate weekday averages per user, movie and globally. (Use APT2 to set parameters.)
- BasicSVD: No more discussion.
- SVD Adaptive User Factors (SVD-AUF) and SVD Alternating Least Squares (SVD-ALS): Both are from BellKor. No more discussion.

Algorithms (continue)

- Weekday Model (WDM): Predict ratings on

All parameters are set by APT2.

$$\bar{r}_{ui} = \frac{\bar{\mu}_{uw} \cdot n_{uw}^{\nu}}{n_{uw}^{\nu} + \alpha} \tag{1}$$

$$\tilde{r}_{ui} = \frac{\bar{\mu}_{iw} \cdot n_{iw}^{\epsilon} + \bar{r}_{ui} \cdot \beta}{n_{iw}^{\epsilon} + \beta} \tag{2}$$

$$\hat{r}_{ui} = \frac{\bar{\mu}_{w} \cdot n_{w}^{\delta} + \tilde{r}_{ui} \cdot \gamma}{n_{w}^{\delta} + \gamma} \tag{3}$$

Both are from BellKor. No more discussion.

Algorithms (continue)

- TimeSVD : Divide the rating time span into T time slots per user, a slot could be a several-day period
- Neighborhood Aware Matrix Factorization (NAMF)
- Restricted Boltzmann Machine (RBM)
- Movie KNN (Neighborhood Model)

Algorithms (continue)

- Regression on Similarity (ROS)
- Asymmetric Factor Model (AFM): From BellKor. No more discussion.
- Global Effects (GE), Global Time Effect (GTE) & Time Dep Model
- Neural Network (NN) & NN Blending (NNBlend)

GE, GTE & TimeDep Model

- GE: One effect could be trained on the residual of previous effect.
- GTE: GE with time dependency.
- TimeDep: An overtime changing rating of a user.
- These are all biases, need to be removed.

GE, GTE & TimeDep Model

Global Effects (GE)

Nr.	Name	RMSE probe	RMSE train	α
0	Overall mean	1.1296	1.0845	NA
1	Movie effect	1.0526	1.0105	22
2	User effect	0.9840	0.9176	7.5
3	User x Time(user)	0.9802	0.9110	435
4	User x Time(movie)	0.9778	0.9060	125
5	Movie x Time(movie)	0.9760	0.9041	4100
6	Movie x Time(user)	0.9752	0.9032	420
7	User x Average(movie)	0.9711	0.8938	68
8	User x support(movie)	0.9682	0.8854	76
9	Movie x average(user)	0.9671	0.8842	140
10	Movie x support(user)	0.9659	0.8836	1e10
11	Movie x avgMovieProductionYear(user)	0.9635	0.8802	380
12	User x movieProductionYear(user)	0.9623	0.8762	170
13	User x standardDeviation(movie)	0.9611	0.8725	130
14	Movie x standardDeviation(user)	0.9604	0.8717	3700

GE, GTE & TimeDep Model

Global Time Effects (GTE)

Nr.	Name	RMSE probe	RMSE train	λ	σ	ϵ	α	β
0	global Avg	1.1271	1.0785	4.67e-4	4.81e-1	1.38e-4	NA	4.52e-8
1	movie Effect	1.0473	1.0033	2.29e1	1.69e1	1.28e-2	1e3	1.25e1
2	user Effect	0.9717	0.9074	2.76e-1	1.32	1.70e-1	8.94	6.91
2.1	movie Effect	0.9705	0.9057	1.01e1	2.40e1	4.07e-5	1.47e4	1.04e-7
2.2	user Effect	0.9686	0.9057	1.62	8.23e1	1.14e-2	1.15e1	1.14e2
3	user x time(user)	0.9679	0.9049	1.37e2	1.33e4	2.54e-3	9.36e-1	3.66e1
4	user x time(movie)	0.9666	0.9014	3.52	2.46e2	3.35e-4	1.06e-2	1.56e2
5	movie x time(movie)	0.9665	0.9012	1.27	2.07e1	6.3e-13	2.08e1	6.72e-8
6	movie x time(user)	0.9653	0.9002	1.88e2	6.23e1	7e-2	9.5	2.27e1
7	user x avg(movie)	0.9619	0.8919	2.70e-3	1.89e2	4.35e-4	1.07e-2	8.94e1
8	user x support(movie)	0.9601	0.8860	4.29e9	1.70e2	1.98e-4	2.05e-3	9.86e1
9	movie x avg(user)	0.9591	0.8852	1.23	1.43e5	1.95e-4	4.58e1	8.50e-2
10	movie x support(user)	0.9581	0.8846	1.12e1	3.06e4	9.18e-3	4.1e-2	1.31
11	movie x avgMovieYear(user)	0.9554	0.8815	7.12e3	3.96e1	1.13e-6	4.1e-4	2.05
12	user x year(movie)	0.9541	0.8775	2.51e2	9.58e3	3.13e-4	1.56	1.25e2
13	user x stddev(movie)	0.9530	0.8739	4.81e-1	6.62e1	1.91e-2	4.1e-3	6.37e1
14	movie x stddev(user)	0.9523	0.8731	9.01e1	1.5e1	4.27e-3	3.56e-6	4.96e-8
15	movie x percentSingleVotes (user)	0.9515	0.8725	7.8e-12	1.59e1	2.31e-8	3.8e-2	7.19e2
16	movie x ratingDateDensity (user)	0.9514	0.8724	3.3e6	9.46	8.6e-11	3.36e1	7.19e-1
17	user x stringlengthMovie Title(movie)	0.9513	0.8713	4.99e4	1.43e2	2.04e-3	2.45e-4	3.14e-6
18	movie x avgStringlenTitle (user)	0.9510	0.8708	6.33e4	4.54e1	3.32e-1	1.09	4.1e-2
19	movie x percentMovieWith NumberInTitle(user)	0.9509 (qual 0.9450)	0.8710	1.92e2	2.53e1	1.03e-4	3.15e-1	3.12

Movie KNN

- Similarity:
  - Movie-based or customer-based.
  - Customer-based impractical; movie-based could be precomputed.
- Best similarities:
  - Pearson Correlation.
  - Set Correlation: $\rho_{ij} = \frac{|N(i) \bigcap N(j)|}{min(|N(i)|,|N(j)|)}$
  - Variable definition: $c_{ij} = \frac{\rho_{ij} \cdot n_{ij}}{n_{ij} + \alpha}$

    α range from 200 to 9000, set by APT1

Movie KNN (continue)

- Basic Pearson KNN (KNN-Basic):
  Simplest form of a KNN model. Weight the K best correlating neighbors based on their correlation c_{ij}.

$$\hat{r}_{ui} = \frac{\sum_{j \in N(u,i)} c_{ij} r_{uj}}{\sum_{j \in N(u,i)} c_{ij}}$$

- KNNMovie
  Extension of basic model. Use sigmoid function to rescale the correlations c_{ij} to achieve lower RMSE.

$$c_{ij}^{new} = \sigma(\delta \cdot c_{ij} + \gamma)$$

$$\sigma(x) = \frac{1}{1 + exp(-x)}$$

$$\hat{r}_{ui} = \frac{\sum_{j \in N(u,i)} c_{ij}^{new} r_{uj}}{\sum_{j \in N(u,i)} c_{ij}^{new}}$$

Movie KNN (continue)

- KNNMovieV3
  Basic idea: give recent ratings a higher weight than the old ones.

$$c_{ij}^{date} = \sigma\left(\delta \cdot c_{ij} \cdot \exp\left(\frac{-\mid \Delta t \mid}{\beta}\right) + \gamma\right)$$

$$\sigma(x) = \frac{1}{1 + exp(-x)}$$

$$\hat{r}_{ui} = \frac{\sum_{j \in N(u,i)} c_{ij}^{date} r_{uj}}{\sum_{j \in N(u,i)} c_{ij}^{date}}$$

- KNNMovieV6
  Not use Pearson or Set correlations. Use the length of common substring between movies and production year to get weighting coefficients.

$$c_{ij}^{sub} = \sigma(\delta \cdot s_{ij}^{\xi} + \gamma) \cdot \exp\left(\frac{-(d_i - d_j)^2}{\beta}\right)$$

$$\sigma(x) = \frac{1}{1 + exp(-x)}$$

$$\tilde{r}_{ui} = \frac{\sum_{j \in N(u)} c_{ij}^{sub} r_{uj}}{\sum_{j \in N(u)} c_{ij}^{sub}}$$

$$\hat{r}_{ui} = \frac{\tilde{r}_{ui} \cdot \sum_{j \in N(u)} c_{ij}^{sub} + \vartheta \cdot \bar{r}_i}{\sum_{j \in N(u)} c_{ij}^{sub} + \vartheta}$$

NAMF

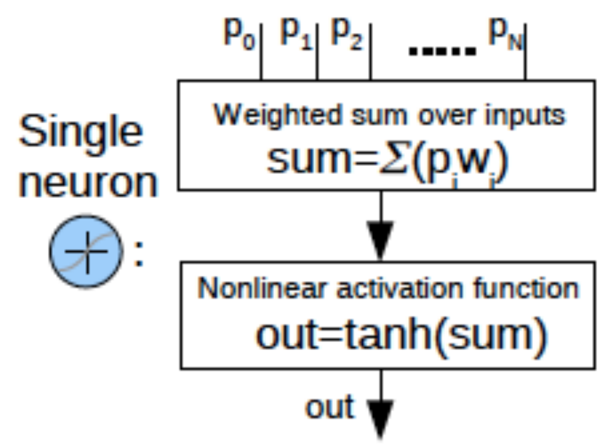
- Key ideas:
  - Combination of matrix factorization and user/item neighborhood models
  - Neighborhood models work best with good correlations
  - The ratings of the best correlating users/items are generally not known
  - Use predicted ratings for the unknown ratings

NAMF (continue)

- Steps:
  - Precompute J-best item and J-best user neighbors for every item/user
  - Train a matrix factorization (RMF)
  - Rating prediction r_{ui} with NAMF
    - Predict r_{ui} directly by trained RMF
    - Predict U_J (u) (J-best user neighbors)
    - Predict I_J (i) (J-best item neighbors)
    - Mix the predictions to get the final prediction for r_{ui}

NN

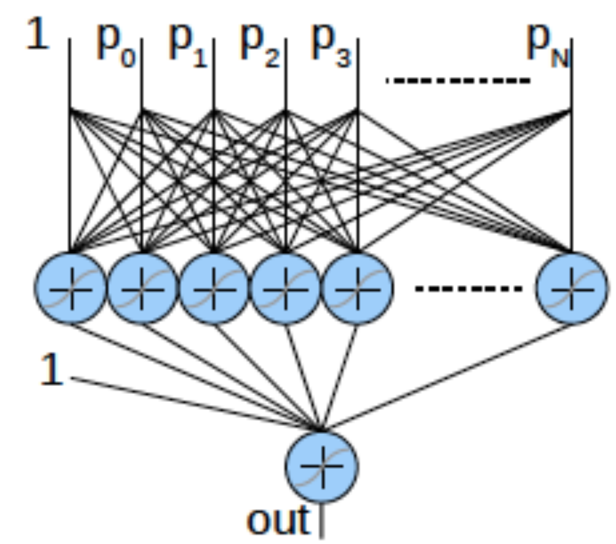
- Single Neuron:
  Take the dot product of input vector p and weight vector w (sometimes with a bias value b).
  Take the dot product as input of activation function to get the output.



- Neural Network:
  Use many neurons to compute, Each neuron needs to be trained to get better weight vector and bias.

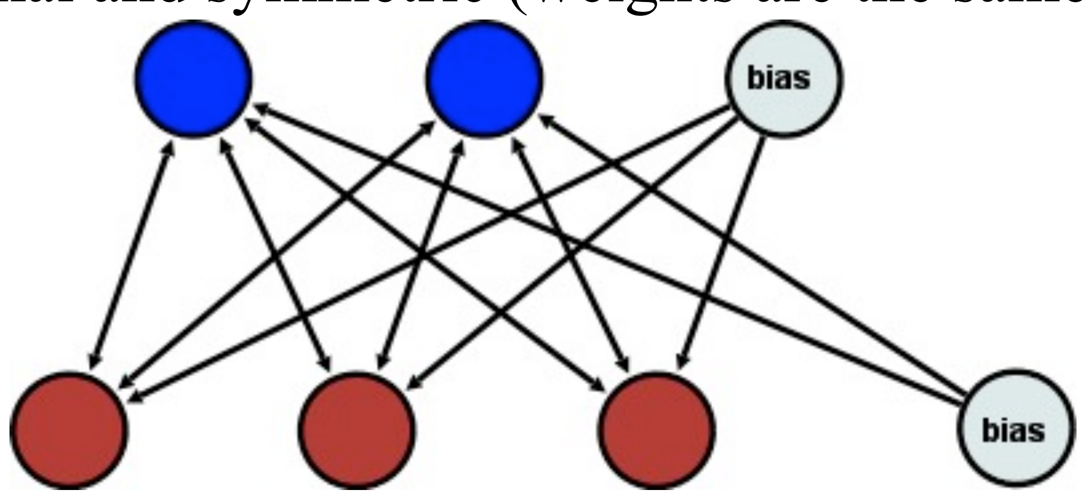
NN (continue)

- Neural Networks (implement):
  - Could have many layers.
  - *M* neurons in the same layer could produce a new vector as the input of next layer.
  - Useful to blend all predictors.
  - Nonlinear works better than linear.



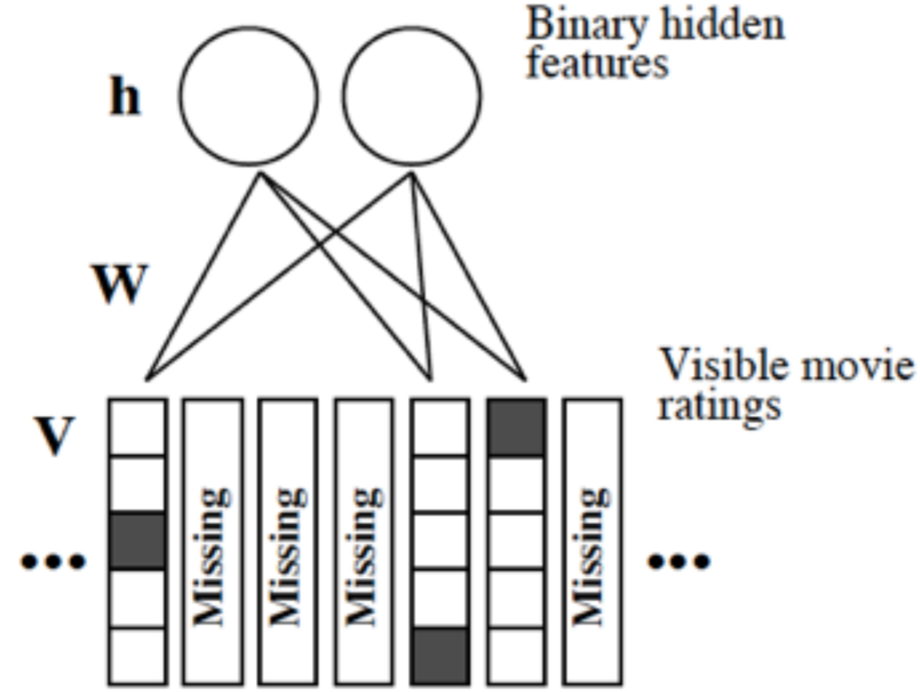
RBM

- From Boltzmann distribution: At thermal equilibrium, energy would be around the global minimum.
- RBM is a stochastic NN (in which each neuron have some random behavior when activated).
    - One visible and one hidden layer; No connection between units in same layer.
    - Each unit connected to all units in other layer. Connections are bidirectional and symmetric (weights are the same in both directions).



RBM (continue)

- RBM used in CF:
  - An RBM with binary hidden units and softmax visible units.
  - The RBM only includes softmax units for the movies that has rated for each user.



  - Biases exist in symmetric weights and each unit.

RBM (continue)

- Equations:
  - Conditional multinomial distribution for modeling each column of visible binary rating matrix V and conditional Bernoulli distribution for hidden user features h:

$$p(v_i^k = 1|\mathbf{h}) = \frac{\exp\left(b_i^k + \sum_{j=1}^F h_j W_{ij}^k\right)}{\sum_{l=1}^K \exp\left(b_i^l + \sum_{j=1}^F h_j W_{ij}^l\right)} \quad (1)$$

$$p(h_j = 1|\mathbf{V}) = \sigma\left(b_j + \sum_{i=1}^m \sum_{k=1}^K v_i^k W_{ij}^k\right) \quad (2)$$

with: $\sigma(x) = 1/(1+e^{-x})$

  - The marginal distribution over the visible ratings V:

$$p(\mathbf{V}) = \sum_{\mathbf{h}} \frac{\exp\left(-E(\mathbf{V},\mathbf{h})\right)}{\sum_{\mathbf{V}',\mathbf{h}'} \exp\left(-E(\mathbf{V}',\mathbf{h}')\right)} \quad (3)$$

  - Energy term: $E(\mathbf{V},\mathbf{h}) = -\sum_{i=1}^m \sum_{j=1}^F \sum_{k=1}^K W_{ij}^k h_j v_i^k + \sum_{i=1}^m \log Z_i - \sum_{i=1}^m \sum_{k=1}^K v_i^k b_i^k - \sum_{j=1}^F h_j b_j \quad (4)$

End-Game

- June 26th 2009:
  Team "BellKorPragmaticChaos" submit 1st 10% better result, trigger 30-day "last call".
- Ensemble team formed: Other leading teams form a new team, combine their models and quickly get 10% better result.
- Before the deadline, both teams kept monitoring the leaderboard, optimizing their algorithms and submitting results once a day.

End-Game (continue)

- Final Results:
  "BellKor" submits a little early, 40 mins before deadline; "Ensemble" submits 20 mins later
- Leaders on test set are contacted and submit their code and documentation (mid-August).
- Judges review documentation and inform winners that they have won \$1 million prize (late August)

End-Game (continue)

Netflix Prize: View Leaderboard

http://www.netflixprize.com//leaderboard?showtest=f&limit=20

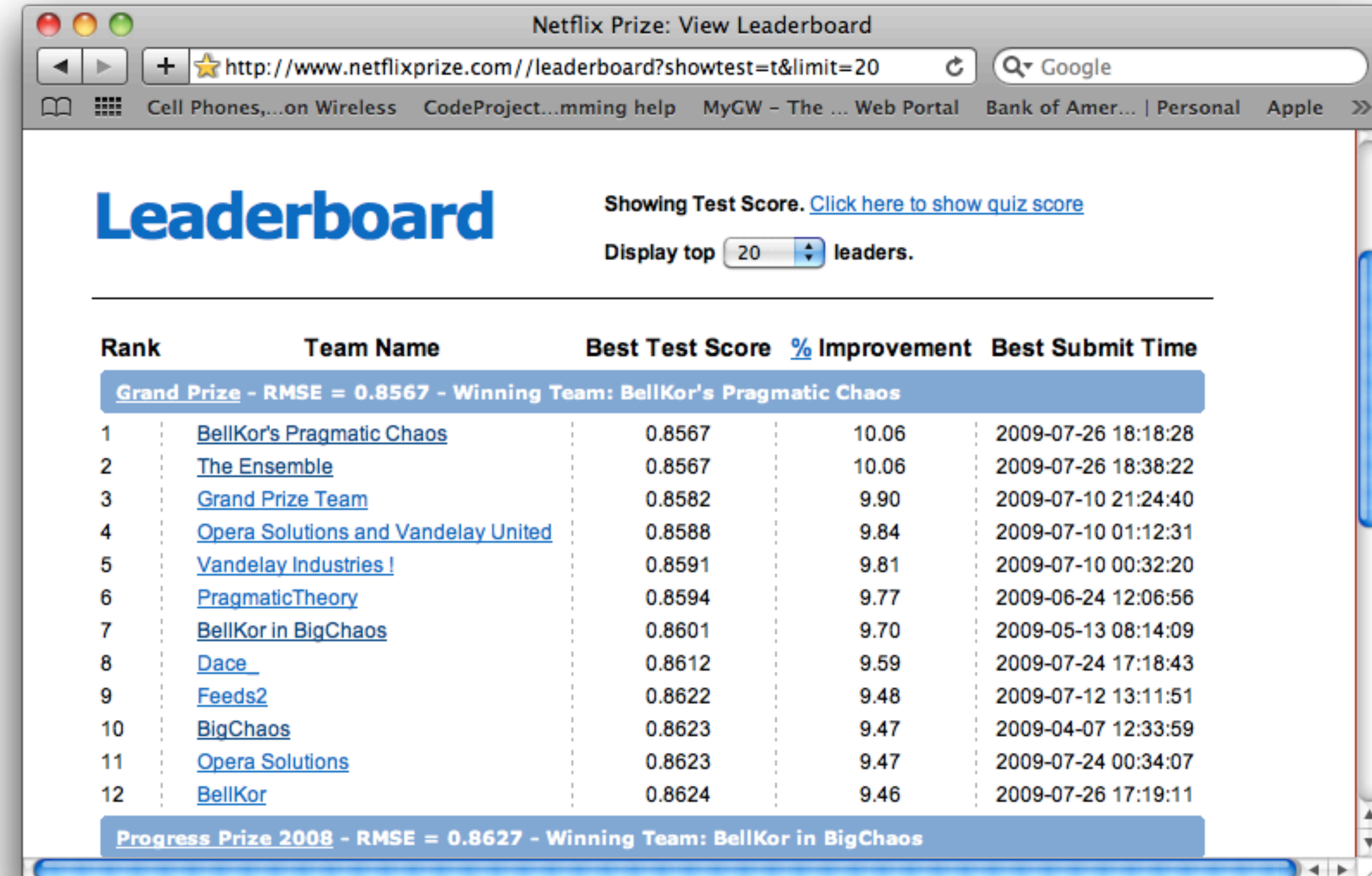
Leaderboard

Showing Quiz Score. Click here to show test score

Display top 20 **leaders.**

Rank	Team Name	Best Quiz Score	% Improvement	Best Submit Time
1	The Ensemble	0.8553	10.10	2009-07-26 18:38:22
Grand Prize - RMSE = 0.8554 - Winning Team: BellKor's Pragmatic Chaos				
2	BellKor's Pragmatic Chaos	0.8554	10.09	2009-07-26 18:18:28
3	Grand Prize Team	0.8571	9.91	2009-07-10 21:24:40
4	Opera Solutions and Vandelay United	0.8573	9.89	2009-07-10 01:12:31
5	Vandelay Industries !	0.8579	9.83	2009-07-10 00:32:20
6	PragmaticTheory	0.8582	9.80	2009-06-24 12:06:56
7	BellKor in BigChaos	0.8590	9.71	2009-05-13 08:14:09
8	Dace	0.8603	9.58	2009-07-24 17:18:43
9	Opera Solutions	0.8611	9.49	2009-07-24 00:34:07
10	BellKor	0.8612	9.48	2009-07-22 20:30:30
11	BigChaos	0.8613	9.47	2009-04-07 12:33:59
12	Feeds2	0.8613	9.47	2009-07-12 13:11:51
Progress Prize 2008 - RMSE = 0.8616 - Winning Team: BellKor in BigChaos				

End-Game (continue)



Netflix Prize: View Leaderboard

http://www.netflixprize.com//leaderboard?showtest=t&limit=20

Cell Phones,...on Wireless CodeProject...mming help MyGW – The ... Web Portal Bank of Amer... | Personal Apple

Leaderboard

Showing Test Score. Click here to show quiz score

Display top 20 **leaders.**

Rank	Team Name	Best Test Score	% Improvement	Best Submit Time
Grand Prize - RMSE = 0.8567 - Winning Team: BellKor's Pragmatic Chaos				
1	BellKor's Pragmatic Chaos	0.8567	10.06	2009-07-26 18:18:28
2	The Ensemble	0.8567	10.06	2009-07-26 18:38:22
3	Grand Prize Team	0.8582	9.90	2009-07-10 21:24:40
4	Opera Solutions and Vandelay United	0.8588	9.84	2009-07-10 01:12:31
5	Vandelay Industries !	0.8591	9.81	2009-07-10 00:32:20
6	PragmaticTheory	0.8594	9.77	2009-06-24 12:06:56
7	BellKor in BigChaos	0.8601	9.70	2009-05-13 08:14:09
8	Dace_	0.8612	9.59	2009-07-24 17:18:43
9	Feeds2	0.8622	9.48	2009-07-12 13:11:51
10	BigChaos	0.8623	9.47	2009-04-07 12:33:59
11	Opera Solutions	0.8623	9.47	2009-07-24 00:34:07
12	BellKor	0.8624	9.46	2009-07-26 17:19:11
Progress Prize 2008 - RMSE = 0.8627 - Winning Team: BellKor in BigChaos				

End-Game (continue)

Conclusion

- From the team “BigChaos”:
  Training and optimizing predictors individually is not optimal.The whole ensemble need to have the right tradeoff between diversity and accuracy. (As Greedy method, local optimal is not global optimal.)
- From the results:
  Collaboration among participants is good.
  Combining models works surprisingly well. (But final 10% improvement can probably be achieved by combining about 10 models rather than 1000’s.)

References

- A.Toscher and M.Jahrer. The BigChaos Solution to the Netflix Prize 2008.
- A. Toscher, M. Jahrer, R. Bell. The BigChaos Solution to the Netflix Grand Prize. 2009.
- R. Salakhutdinov, A. Mnih, and G. E. Hinton. Restricted Boltzmann machines for collaborative filtering. In ICML, pages 791-798, 2007.
- A. Toscher, M. Jahrer, and R. Legenstein. Improved neighborhood-based algorithms for large-scale recommender systems. In KDD Workshop at SIGKDD 08, August 2008.
- Padhraic Smyth. Netflix Competition Overview . Lecture note of CS 277: Data Mining, download from http://www.ics.uci.edu/~smyth/courses/cs277/slides/netflix_overview.pdf

Q & A