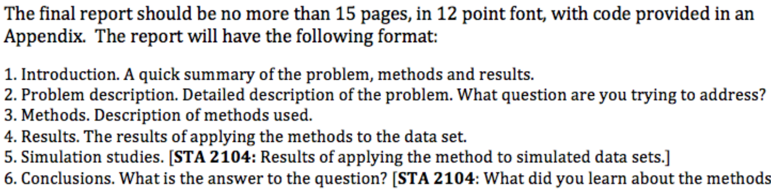
Notes

- no class Thursday, April 2
- project due **Thursday, April 16 before 2 pm**
- see `http://www.utstat.utoronto.ca/reid/sta414/414S10-html.doc` for outline (Jan 5,7)



The final report should be no more than 15 pages, in 12 point font, with code provided in an Appendix. The report will have the following format:

1. Introduction. A quick summary of the problem, methods and results.
2. Problem description. Detailed description of the problem. What question are you trying to address?
3. Methods. Description of methods used.
4. Results. The results of applying the methods to the data set.
5. Simulation studies. [**STA 2104:** Results of applying the method to simulated data sets.]
6. Conclusions. What is the answer to the question? [**STA 2104**: What did you learn about the methods

- Take-home MT graded by Tuesday, April 6;
  pickup in SS 6003

The Netflix Grand Prize

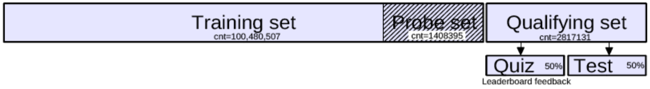
- "The BellKor Solution to the NGP", Koren (August 2009)
- "The BigChaos Solution to the NGP", Töscher, Jahrer, Bell (September 2009)
- "The Pragmatic Theory solution to the Netflix Grand Prize", M. Piotte, M. Chabbert, (August 2009).
- "The BellKor 2008 Solution to the Netflix Prize", Bell, Koren, Volinsky
- "All Together Now: A Perspective on the Netflix Prize", Bell, Koren, Volinsky (2010) in *Chance* **23**, 24 – 29.

- first four papers available from `http://www.netflixprize.com//index`

NETFLIX
2009
DATE 09 21 09
PAY TO THE ORDER OF BellKor's Pragmatic Chaos
$ 1,000,000 00
AMOUNT ONE MILLION 00/100
FOR The Netflix Prize

The data

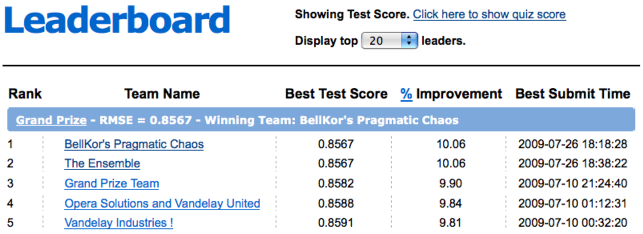
- approx 18,000 movies; nearly 500,000 users
- total approx 100 million ratings, collected over seven years
- ratings are 1 to 5 stars
- collaborative filtering models: use models built on the training data to make individualized predictions
- hold-out set of approx 4.2 million ratings
- split into three subsets: probe set, quiz set, test set



- prizemaster reports value of root-mean-squared-error (RMSE) for the quiz set on the leaderboard
- winner determined by the first to improve RMSE on the test set by 10% beyond Netflix's Cinematch system

The contest

- Cinematch RMSE at the beginning of the contest: 0.9525
- benchmark to win: 90% of this: 0.8572
- winning entries: 0.8567
  `http://www.netflixprize.com//leaderboard`


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

Some wrinkles

- hold-out set (probe, quiz, test) are last nine movies rated by each user, (or fewer)
- contains many more ratings by users that do not rate much
- harder to predict
- some users rated more than 10,000 movies
- average number of ratings per user is 208
- 25% of users rated fewer than 50 movies
- if we view ratings/users as a huge matrix r_{ui}: 99% of the entries are missing
- training set is $\mathcal{T} = \{(u, i) \mid r_{ui} \text{is known}\}$
- $R(u)$: all the items rated by user u
- $R(i)$: all the users who rated movie i
- $N(u)$: all the items for which u provided a rating, even if the rating is unknown (i.e. qualifying set)

General strategy

- avoid overfitting by regularizing parameter estimates
- i.e. penalizing $||\theta||^2$ ("*L*2 regularization")
- when a new tuning constant introduced, choose best RMSE on probe set over several runs
- and keep that tuning constant fixed going forward
- BigChaos introduced a second training run with the chosen tuning constant, using probe and training data
- since various predictors will be aggregated, tuning could be chosen based on aggregation

Baseline predictors

- $$b_{ui} = \mu + b_u + b_i$$
- some users are more (or less) critical than average
- some movies are more (or less) popular than average
- $$\min_b \sum_{(u,i)\in\mathcal{T}} (r_{ui} - \mu - b_u - b_i)^2 + \lambda_3\Big(\sum_u b_u^2 + \sum_i b_i^2\Big)$$
- movies' popularity changes over time
- users' ratings change over time: either drift, or because users in a household change
- $$b_{ui} = \mu + b_u(t_{ui}) + b_i(t_{ui})$$

... finetuning the baseline

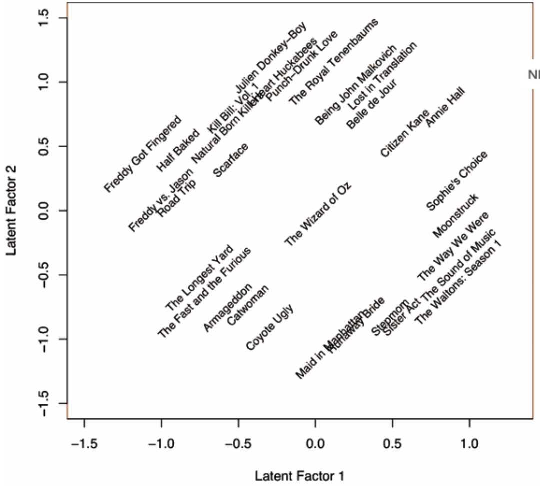
- $b_{ui} = \mu + b_u(t_{ui}) + b_i(t_{ui})$
- $b_i(t) = b_i + b_{i,\mathrm{Bin}(t)}$
- Bin(t) is the 10-week period that time t falls in
- 30 such Bins span the time of the dataset $b_i(t) = b_i + b_{i,j_t}$
- $$b_u(t) = b_u + \alpha_u \mathrm{dev}_u(t) + b_{ut}$$
- $\mathrm{dev}_u(t) = \mathrm{sign}(t - t_u)|t - t_u|^\beta$
- t_u: mean rating date for user u
- β: 0.4 (tuning on probe set)
- α_u, b_u user-specific parameters
- b_{ut}: single parameter per user per day (approximately 40 parameters per user)

... finetuning the baseline

- $$b_{ui} = \mu + b_u + \alpha_u \text{dev}_u(t_{ui}) + b_{u,t_{ui}} + b_i + b_{i,\text{Bin}(t_{ui})}$$
- movie bias is not user-independent: users have different rating scales, and a single user changes over time
- $$b_{ui} = \mu + b_u + \alpha_u \text{dev}_u(t_{ui}) + b_{u,t_{ui}} + (b_i + b_{i,\text{Bin}(t_{ui})}) c_u(t_{ui})$$
- $c_u(t) = c_u + c_{ut}$
- RMSE 0.9555 (Cinematch 0.9514)
- frequencies of rating: F_{ui}: number of ratings user u gave on day t_{ui}; $f_{ui} = \log F_{ui}$
- $$b_{ui} = \mu + b_u + \alpha_u \text{dev}_u(t_{ui}) + b_{u,t_{ui}} + (b_i + b_{i,\text{Bin}(t_{ui})}) c_u(t_{ui}) + b_{i,f_{ui}}$$
- RMSE 0.9278

Adding to the baseline: Matrix factorization

- characterize users and movies by d-vectors of latent factors: $d = 20, 200, 500, 1000, 2000$
- $\hat{r}_{ui} = \mu + b_i + b_u + q_i^T p_u$ [1]



... matrix factorization

- $$\hat{r}_{ui} = \mu + b_i + b_u + q_i^T p_u$$
- $$p_{uk}(t) = p_{uk} + \alpha_{uk} \mathrm{dev}_u(t) \quad k = 1, \ldots d$$
- $$q_i^T \to q_i^T + q_{i,f_{ui}}^T$$
- $$\hat{r}_{ui} = b_{ui} + (q_i^T + q_{i,f_{ui}}^T) \left(p_u(t_{ui}) + |N(u)|^{-1/2} \sum_{j \in N(u)} y_j \right)$$
- RMSE 0.8777 ($d = 200$)

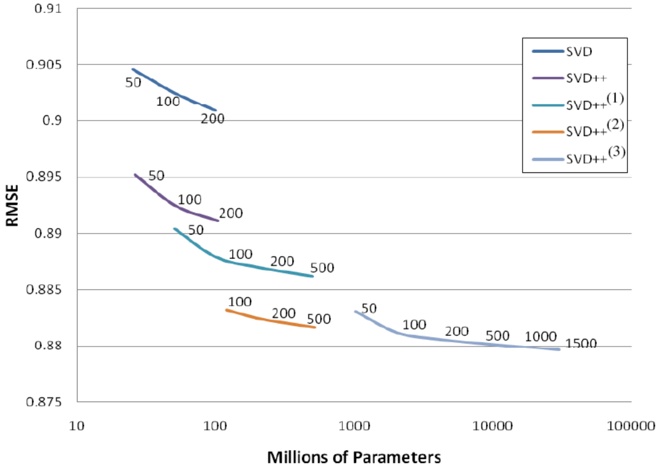


Figure 1. Matrix factorization models – error vs. #parameters. The plot shows how the accuracy of each of five individual factor models improves by increasing the

Adding to the baseline: Neighbourhood models

- simple version:

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

- $N(i, u)$ the set of neighbours of movie i rated by user u
- s_{ij} similarity between items i and j
- Bellkor version $\hat{r}_{ui} =$ baseline $+$ user-item interaction
- $$\hat{r}_{ui} = b_{ui} + |R(u)|^{-1/2} \sum_{j \in R(u)} w_{ij}(r_{uj} - \tilde{b}_{uj}) + |N(u)|^{-1/2}| \sum_{j \in N(u)} c_{ij}$$
- $R(u)$: all the items rated by user u in the training data
- $N(u)$: all the items for which there is a rating by user u (those in qualifying set unknown)
- w_{ij}, c_{ij} unknown parameters to be estimated
- "adjustments we need to make to the predicted rating of item i, given a rating of item j"

Adding time to neighbourhood models

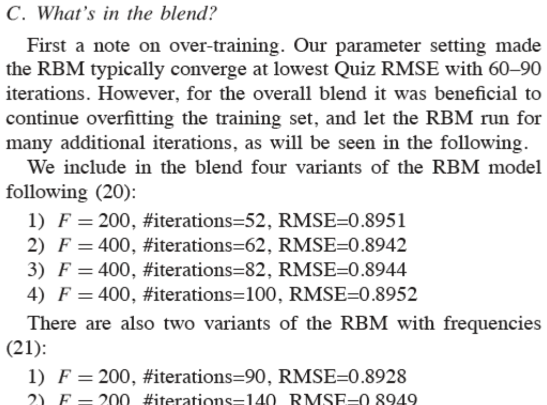
- $$\hat{r}_{ui} = b_{ui} + |R(u)|^{-1/2} \sum_{j \in R(u)} w_{ij}(r_{uj} - \tilde{b}_{uj}) e^{-\beta_u |t_{ui} - t_{uj}|}$$
$$+ |N(u)|^{-1/2}| \sum_{j \in N(u)} c_{ij} e^{-\beta_u |t_{ui} - t_{uj}|}$$
- RMSE 0.8870
- another variant:

$$\hat{r}_{ui} = b_{ui} + |R^k(i;u)|^{-1/2} \sum_{j \in R^k(i;u)} w_{ij}(r_{uj} - \tilde{b}_{uj}) e^{-\beta_u |t_{ui} - t_{uj}|}$$
$$+ |N^k(i;u)|^{-1/2}| \sum_{j \in N^k(i;u)} c_{ij} e^{-\beta_u |t_{ui} - t_{uj}|}$$

- $N^k(i;u)$: k neighbours for movie i

Other approaches also used

- RBM: Restricted Boltzmann Machines
  Salakhutdinov, R. R., Mnih, A. and Hinton, G. E., 2007
- roughly, a complex neural network, adapted to matrix data
- implemented by BellKor and by Pragmatic Theory
- fancy bits added to account for time



C. What's in the blend?

First a note on over-training. Our parameter setting made the RBM typically converge at lowest Quiz RMSE with 60–90 iterations. However, for the overall blend it was beneficial to continue overfitting the training set, and let the RBM run for many additional iterations, as will be seen in the following.

We include in the blend four variants of the RBM model following (20):

1) $F = 200$, #iterations=52, RMSE=0.8951
2) $F = 400$, #iterations=62, RMSE=0.8942
3) $F = 400$, #iterations=82, RMSE=0.8944
4) $F = 400$, #iterations=100, RMSE=0.8952

There are also two variants of the RBM with frequencies (21):

1) $F = 200$, #iterations=90, RMSE=0.8928
2) $F = 200$, #iterations=140, RMSE=0.8949

... other approaches

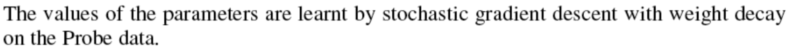
- Gradient Boosting Decision Trees: Big Chaos
- used for blending predictors developed above
- "... the final 30 days of the competition, we ... speculated that the most impact ... would be achieved by exploring new blending techniques"
- decision tree: uses predictions from earlier models as input; construct a regression model for the output
- boosted: an ensemble of trees fitted in a forward step-wise manner
- most trees are "shallow" (few splits)
- example: 24 predictors; 150 trees, tree-size 20 RMSE 0.8664
- example: 454 predictors; 200 trees, tree-size 20 RMSE 0.8603

Alternate blending Progress Prize 2008, 2007

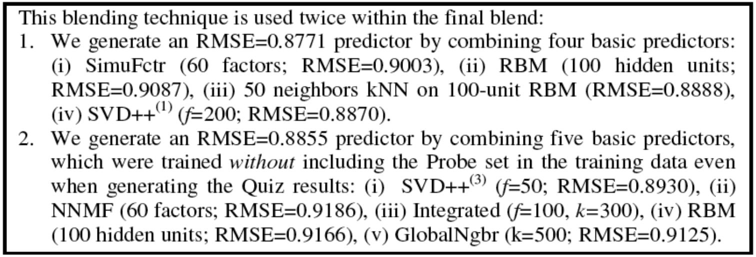
- given a series of s predictors $r^{(k)} = \{r_{ui}^{(k)}\}, k = 1, \dots s$
  really $\hat{r}_{ui}$
- simple linear combination $\hat{r} = \sum_{k=1}^{s} a^{(k)} r^{(k)}$
- estimate a's from the probe set
- let the coefficients also depend on the movie and the user
- $\hat{r}_{ui} = \sum_{k=1}^{s} (a^{(k)} + b_i^{(k)} + c_u^{(k)}) r_{ui}^{k}$
- too many coefficients for users, probe set has a small number of users
- $$\hat{r}_{ui} = \sum_{k=1}^{s} (a^{(k)} + b_i^{(k)} + c^{(k)} \log |R(u)| + d^{(k)} \log |R(i)|) r_{ui}$$

... alternate blending

- $$\hat{r}_{ui} = \sum^{s} (a^{(k)} + b_i^{(k)} + c^{(k)} \log |R(u)| + d^{(k)} \log |R(i)|) r_{ui}$$



The values of the parameters are learnt by stochastic gradient descent with weight decay on the Probe data.



This blending technique is used twice within the final blend:

1. We generate an RMSE=0.8771 predictor by combining four basic predictors: (i) SimuFctr (60 factors; RMSE=0.9003), (ii) RBM (100 hidden units; RMSE=0.9087), (iii) 50 neighbors kNN on 100-unit RBM (RMSE=0.8888), (iv) SVD++[(1)] (*f*=200; RMSE=0.8870).
2. We generate an RMSE=0.8855 predictor by combining five basic predictors, which were trained *without* including the Probe set in the training data even when generating the Quiz results: (i) SVD++[(3)] (*f*=50; RMSE=0.8930), (ii) NNMF (60 factors; RMSE=0.9186), (iii) Integrated (*f*=100, *k*=300), (iv) RBM (100 hidden units; RMSE=0.9166), (v) GlobalNgbr (k=500; RMSE=0.9125).

neural network versions also used by, e.g., Pragmatic Theory

... fine-tuning the blending BigChaos, p.21

- "Some of our models deliver very good results on the residuals of others. A well-known combination are neighborhood approaches on the residuals of RBMs"
- residuals are from training data, on a model fit to training data $\rightarrow$ too small
- cross-validation residuals used instead "We always use $K = 34$"

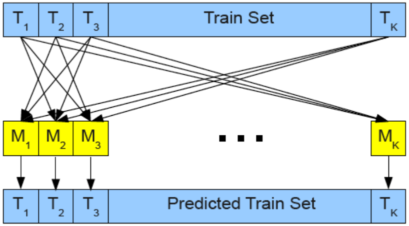


Figure 4: This figure visualizes the idea of "correct residuals". We split the train set into K disjoint sets T_1 to T_K of equal size and train K different CF models M_1 to M_K. The first model M_1 uses the ratings of the sets T_2 to T_K for training and generates predictions for the set T_1. The second model M_2 excludes the set T_2 in the training phase, and calculates predictions for this set. Each rating in the training set is predicted by 1 model. Each rating in the probe and qualifying set is predicted by 34 models. The predictions for the probe and the qualifying set are linear blends of all K models.

Conclusions

- interest in and extensions of research in collaborative filtering (methods and people)
- success of joining groups “crowdsourcing”
- ensemble methods
- benefits of collaboration
- “Out of the numerous new algorithmic contributions, I would like to highlight one – those humble baseline predictors”[2]
- “we have learned that an accurate treatment of main effect is probably at least as significant as ... modelling breakthroughs”
- “will allow the creation of higher accuracy recommendation systems or, at the very least, simpler ones with equivalent accuracy”[3]

[2] BellKor 09

[3] Pragmatic Theory 09

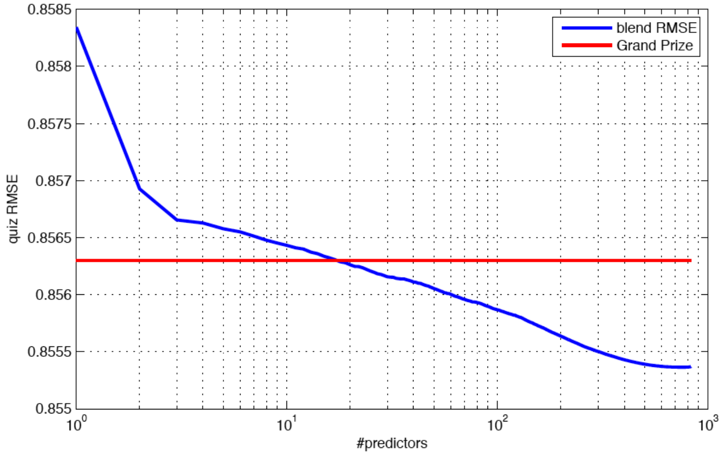


Figure 6: How many results are really needed? With 18 results we breach the 10% (RMSE 0.8563) barrier. Within these results there are 11 nonlinear probe blends and 7 unblended predictors. An ordered list of these predictors can be found in Appendix C.

BigChaos, 2009

Technical Note: “stochastic gradient descent”

- $R(\theta) = \sum_{k=1}^{K} \sum_{i=1}^{N} \{y_{ik} - f_k(x_i)\}^2$ or $-\sum_{k=1}^{K} \sum_{i=1}^{N} y_{ik} \log f_k(x_i)$
- $$\frac{\partial R(\theta)}{\partial \theta} = 0 \quad \text{or} \quad \frac{\partial R(\theta) + \lambda J(\theta)}{\partial(\theta, \lambda)} = 0$$
- $$\theta_j^{(r+1)} = \theta_j^{(r)} + \gamma_r \frac{\partial R(\theta)}{\partial \theta_j^{(r)}}$$
- γ_r “learning rate”: in stochastic version changes with r
- in ordinary (non-stochastic) gradient descent, γ_r is fixed
- learning rate controls step size in direction of decrease
- helps to avoid overfitting to local minima

Example: Neural networks[4]

- $f_i(x_i) = g_k(\beta_{0k} + \beta_k^T z_i)$
- $z_{mi} = \sigma(\alpha_{0m} + \alpha_m^T x_i)$
- $Y_k = g_k\{\beta_{0k} + \sum_{m=1}^M \beta_{km}\sigma(\alpha_{0m} + \sum_{\ell=1}^p \alpha_{\ell m} X_\ell)\} = f_k(X)$ Mar 9
- $$\begin{aligned} \frac{\partial R_i(\theta)}{\partial \beta_{km}} &= -2\{y_{ik} - f_k(x_i)\}g_k'(\beta_k^T z_i)z_{mi} \\ \frac{\partial R_i(\theta)}{\partial \alpha_{m\ell}} &= -2\{y_{ik} - f_k(x_i)\}g_k'(\beta_k^T z_i)\beta_{km}\sigma'(\alpha_m^T x_i)x_{i\ell} \end{aligned}$$
- $\beta_{km}^{(r+1)} = \beta_{km}^{(r)} - \gamma_r \sum_{i=1}^N \frac{\partial R_i(\theta)}{\partial \beta_{km}^{(r)}}$
- $\alpha_{m\ell}^{(r+1)} = \alpha_{m\ell}^{(r)} - \gamma_r \sum_{i=1}^N \frac{\partial R_i(\theta)}{\partial \alpha_{m\ell}^{(r)}}$

[4] pp.396–397 of HTF

Summary

- Regression: linear, ridge, lasso, logistic, polynomial splines, smoothing splines, kernel methods, additive models, regression trees, projection pursuit, neural networks: Chapters 3, 5, 9, 11
- Classification: logistic regression, linear discriminant analysis, generalized additive models, kernel methods, naive Bayes, classification trees, support vector machines, neural networks, K-means, k-nearest neighbours, random forests: Chapters 4, 6, 9, 11, 12, 15
- Model Selection and Averaging: AIC, cross-validation, test error and training error, bootstrap aggregation: Chapter 7, 8.7
- Unsupervised learning: Kmeans clustering, k-nearest neighbours, hierarchical clustering: Chapter 14
- Left out: MARS, boosting, flexible discriminant analysis, mixture discriminant analysis, independent components analysis, multidimensional scaling, graphical models, $p >> N$: Chapters 10, 12, 13, 14, 16, 17, 18

... summary

- Suggestion: Read Chapter 2
- Pattern Recognition and Neural Networks. B.D. Ripley (1996), Cambridge University Press. *Good discussion of many machine learning methods.*
- Classification (2nd ed.), A. D. Gordon (1999), Chapman & Hall/CRC Press. *Unsupervised learning/clustering; see Ch. 2 for good description of dissimilarity measures.*
- Finding Groups in Data: An Introduction to Cluster Analysis, L. Kaufman and P.J. Rousseeuw, (1990) Wiley. *Learn all about daisy, agnes, and many other of R's clustering methods.*
- Modern Applied Statistics with S (4th Ed.), W.N. Venables and B.D. Ripley (2002), Springer-Verlag. *The bible for computing with Splus and R; Ch. 11 covers unsupervised learning, Chs. 8,9 and 12 cover supervised learning.*
- Principles of Data Mining. D. Hand, H. Mannila, P. Smyth (2001) MIT Press. *Nice blend of computer science and statistical methods. Clustering covered in Ch. 9*