

Combining Predictions for Accurate Recommender Systems

[KDD 2010, July 25–28, 2010, Washington D.C., USA]

Michael Jahrer
Andreas Töscher
Robert Legenstein

Outline

- Motivation
- Collaborative Filtering Algorithms
- Blending Algorithms
- Experimental Results on Netflix Data
- Application example: KDD Cup 2010

Motivation

- Accurate recommendations may increase the sales
- Guides users to the products, they want to purchase
- Better cross-selling
- Increasing user activity

The screenshot shows the Amazon.com homepage for user Michael Jahrer. The navigation bar includes the Amazon logo, user name, and links for Today's Deals, Gifts & Wish Lists, and Gift Cards. The search bar is set to 'All Departments'. Below the navigation bar, there are links for 'Your Amazon.com', 'Your Browsing History', 'Recommended For You', 'Rate These Items', 'Improve Your Recommendations', 'Your Profile', 'Your Communities', and 'Learn More'. The main content area is titled 'Recommended for You' and features a recommendation for the book 'Nerds on Wall Street: Math, Machines and Wired Markets' by David J. Leinweber. The book cover is displayed, along with the author's name and the publication date (June 9, 2009). Below the book cover, there are buttons for 'See all buying options' and 'Add to Wish List'. The recommendation is based on items the user owns and more. The page also includes a 'Just For Today' section and a 'Recommendations' sidebar with links to various product categories.

Collaborative filtering

- All algorithms have been successfully applied on the Netflix Prize dataset
 - SVD – Singular Value Decomposition
 - KNN – K-Nearest Neighbors (item - item)
 - AFM – Asymmetric Factor Model
 - RBM – Restricted Boltzmann Machines
 - GE – Global Effects

SVD

u ...user i ...item
 \hat{r}_{ui} ...prediction
 \mathbf{p} ...item feature
 \mathbf{q} ...user feature

- Very popular since the Netflix Prize
- Accurate and good scaling properties

$$\hat{r}_{ui} = \mathbf{p}^T \mathbf{q} = \begin{array}{|c|c|c|} \hline p_0 & p_1 & p_3 \\ \hline \end{array} * \begin{array}{|c|} \hline q_0 \\ \hline q_1 \\ \hline q_2 \\ \hline \end{array}$$

item feature user feature

KNN

u ...user i ...item
 \hat{r}_{ui} ...prediction
 $R(u, i)$...item set
 c_{ij} ...corr between
item i and j
 r_{ui} ...user rating

- Natural approach
- Predict a rating \hat{r}_{ui}
 - Find k -best correlating items $R(u, i)$
 - Make a weighted sum

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

- Quadratic runtime for 1x prediction $O(N^2)$ $N = \#items$

AFM

u ...user i ...item
 \hat{r}_{ui} ...prediction
 $N(u)$...ratings of u
 \mathbf{p} ...item feature
 \mathbf{q} ...asym. item feature

- Like SVD
- A user is represented via his rated items $N(u)$
- ☺ New users can be integrated without re-training

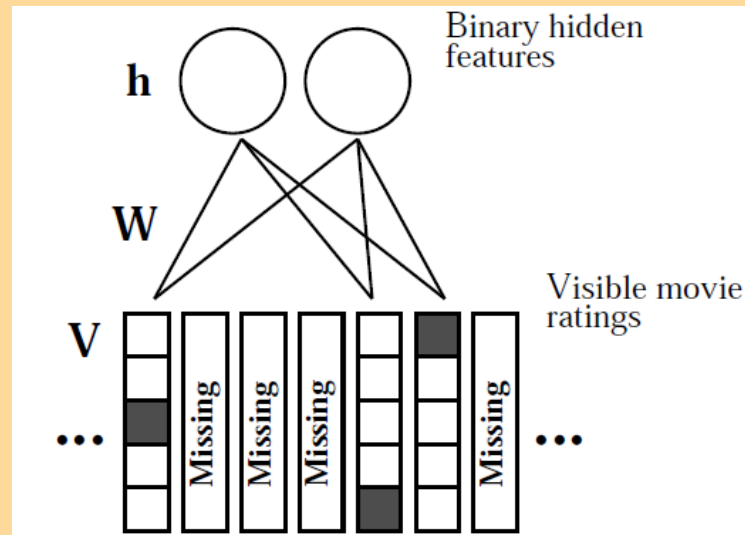
$$\hat{r}_{ui} = \mathbf{p}^T \left(\frac{1}{\sqrt{|N(u)|}} \sum_{j \in N(u)} \mathbf{q}_j \right) = \begin{bmatrix} p_0 & p_1 & p_3 \end{bmatrix} * \frac{1}{\sqrt{3}} \left(\begin{bmatrix} q_0 \\ q_1 \\ q_2 \end{bmatrix} + \begin{bmatrix} q_0 \\ q_1 \\ q_2 \end{bmatrix} + \begin{bmatrix} q_0 \\ q_1 \\ q_2 \end{bmatrix} \right)$$

item feature user feature (virtual)

RBM

u ...user i ...item
 \hat{r}_{ui} ...prediction

- Two-layer undirected graphical model
- Learning is performed with "contrastive divergence"
- RBM reconstructs the visible units
- Predictions \hat{r}_{ui} are calculated over rating probabilities



[R.Salakhutdinov, A.Mnih, G.Hinton : **Restricted Boltzmann machines for collaborative filtering**, ICML '07]

Global Effects

- Calculate "hand-crafted" features for users and items
- Equivalent to SVD with either fixed user or item features

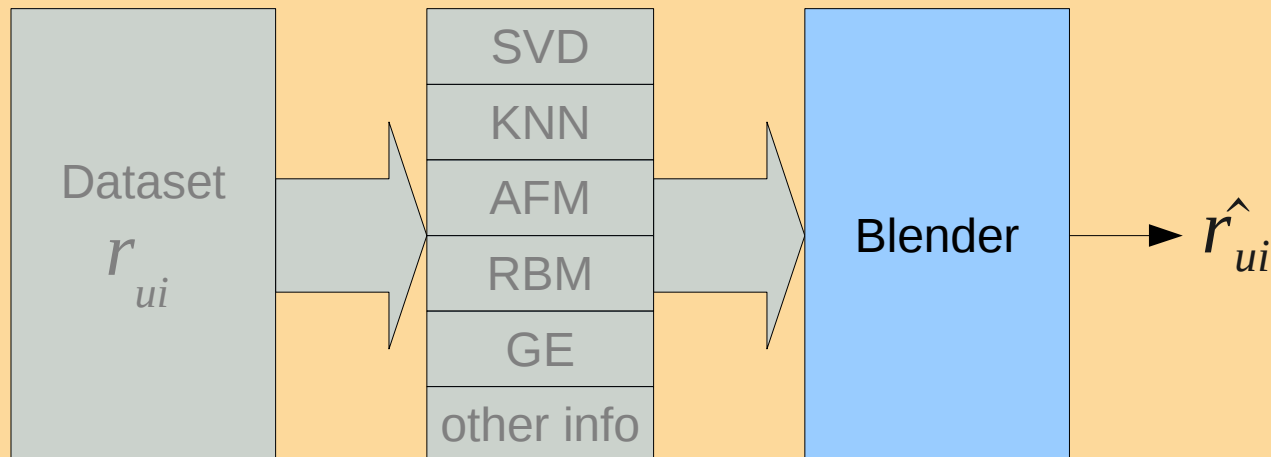
#	effect	shrinkage
1	Movie effect	α_1
2	User effect	α_2
3	User effect: user x sqrt(time(user))	α_3
4	User effect: user x sqrt(time(movie))	α_4
5	Movie effect: movie x sqrt(time(movie))	α_5
6	Movie effect: movie x sqrt(time(user))	α_6
7	User effect: user x average(movie)	α_7
8	User effect: user x votes(movie)	α_8
9	Movie effect: movie x average(user)	α_9
10	Movie effect: movie x votes(user)	α_{10}

[A.Töscher, M.Jahrer, R.Bell : **The BigChaos Solution to the Netix Grand Prize**, 2009]

Blending

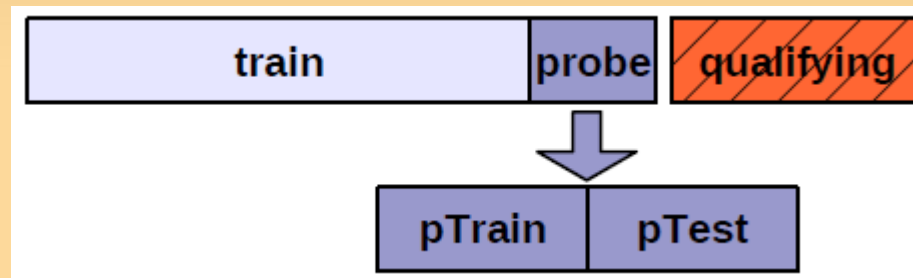
u ...user i ...item
 \hat{r}_{ui} ...prediction
 $N(u)$...ratings of u

- Apply a supervised learner for combining predictions
- Error: RMSE
- Additional information: $|N(u)|$ (the "support")



Evaluation schema

- Dataset for CF algorithms: **Netflix** (10^8 ratings, except probe)
- Dataset for Blending: **probe** (1.4M ratings)
- 50/50 random split of **probe**: **pTrain**, **pTest**



Blending

- **pTrain**: training set
- **pTest**: test set
- **qualifying**: another test set

Used CF algorithms

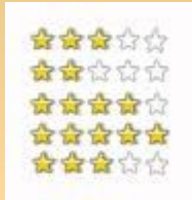
- 4x SVD
- 4x AFM
- 4x KNN
- 2x RBM
- 4x GE
- log(support) as additional input

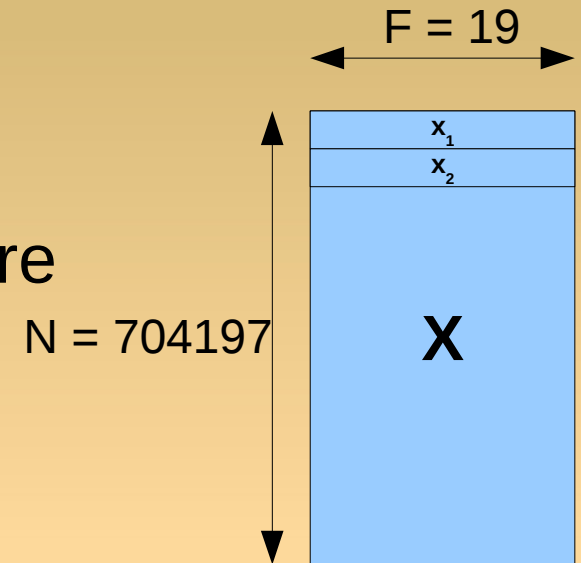
19 predictors

- Some are trained on residuals of others

nr	name	RMSE	description
1	AFM-1	0.9362	AFM, 200 features, $\eta = 1e-3$, $\lambda = 1e-3$, learnrate η is multiplied with 0.95 from epoch 30, trained for 120 epochs
2	AFM-2	0.9231	AFM, 2000 features, $\eta = 1e-3$, $\lambda = 2e-3$, trained for 23 epochs, based on residuals of KNN-4.
3	AFM-3	0.9340	AFM, 40 features, $\eta = 1e-4$, $\lambda = 1e-3$, trained for 96 epochs
4	AFM-4	0.9391	AFM, 900 features, $\eta = 1e-3$, $\lambda = 1e-2$, trained for 43 epochs
5	GE-1	0.9079	GE, 16 effects, based on residuals of KNN-1
6	GE-2	0.9710	GE, 16 effects, on raw ratings
7	GE-3	0.9443	GE, 16 effects, based on residuals of KNN-4
8	GE-4	0.9209	GE(with time), 24 effects, based on residuals of AFM-2
9	KNN-1	0.9110	KNN item, Pearson correlation, $k = 24$ neighbors, based on residuals of AFM-1
10	KNN-2	0.8904	KNN item, Set correlation [20], $k = 122$, based on residuals from a chain of algorithms RBM-KNN-GE(with time)
11	KNN-3	0.8970	KNN item, Pearson correlation, $k = 55$, based on residuals of a discrete RBM model with $nHid = 150$
12	KNN-4	0.9463	KNN item, Pearson correlation, $k = 21$, based on residuals of GE-2
13	RBM-1	0.9493	RBM, discrete, $nHid = 10$, $\eta = 0.002$, $\lambda = 0.0002$
14	RBM-2	0.9123	RBM, discrete, $nHid = 250$, $\eta = 0.002$, $\lambda = 0.0004$
15	SVD-1	0.9074	SVD, 300 features, $\eta = 8e-4$, $\lambda = 0.01$, trained for 158 epochs, based on residuals of 1GE (item mean)
16	SVD-2	0.9172	SVD, 20 features, $\eta = 0.002$, $\lambda = 0.02$, trained for 158 epochs, based on residuals of 1GE (item mean)
17	SVD-3	0.9033	SVD, 1000 features, with adaptive user factors (AUF [20]), $\eta = 0.001$, $\lambda = 0.015$, trained for 158 epochs
18	SVD-4	0.8871	SVD extended, 150 features, individual learnrates η and regularization constants λ are automatically tuned on the probe set [20].
19	support	-	The number of ratings per user; we take the natural logarithm of the support as additional input.

Blending (supervised setup)

- \mathbf{X} ... train set (N x F matrix)
- x_{ij} ... feature value i...sample, j...feature
- \mathbf{y} ... targets (1...5 ratings) 
- \mathbf{p} ... predictions
- $\Omega(\mathbf{x})$... model (the "blender")
- Error function



$$RMSE = \sqrt{\frac{1}{N} \sum_{i=1}^N (\Omega(\mathbf{x}) - y_i)^2}$$

What is inside X ?

(first 20 rows)

(X=train set)

Predictors																			Target
4,12	4,18	3,74	4,08	4,43	4,11	4,10	4,11	4,32	4,61	4,63	4,11	4,28	4,53	4,67	4,53	4,20	4,18	4,55	5
3,62	3,79	3,73	3,81	3,60	3,68	3,76	3,76	3,62	2,72	2,75	3,69	3,62	2,43	2,26	1,68	3,12	3,16	6,18	1
3,22	3,22	3,55	3,59	3,05	3,67	3,48	3,28	3,22	3,16	3,17	3,70	4,04	3,10	3,32	3,74	2,94	2,87	4,30	1
3,35	4,15	3,26	3,69	3,60	2,99	4,09	4,16	3,52	3,94	3,98	4,17	4,59	3,71	4,23	4,18	3,68	3,97	4,06	4
4,11	3,93	3,84	4,01	3,94	3,58	3,85	3,83	4,04	3,81	3,94	3,89	3,82	3,94	3,82	3,80	3,73	3,86	5,82	3
4,22	4,55	4,09	4,20	4,42	4,15	4,38	4,55	4,45	4,19	4,21	4,56	4,15	4,23	4,24	4,24	4,15	4,26	4,49	4
3,64	3,93	3,65	3,59	3,82	3,62	3,96	3,98	3,82	3,86	3,80	3,74	3,51	3,77	3,91	4,24	3,83	4,00	5,09	3
2,50	2,61	2,60	2,64	2,34	2,84	2,75	2,61	2,29	2,32	2,32	2,86	3,10	2,63	2,57	2,47	2,57	2,75	5,49	1
3,44	2,98	3,30	3,55	3,32	3,13	3,09	3,15	3,45	3,10	3,27	3,08	3,38	3,00	3,39	3,34	2,99	3,32	3,43	4
3,75	3,91	3,73	3,82	3,90	3,97	3,94	3,90	3,92	3,80	3,76	4,02	3,84	3,98	4,29	4,30	3,98	4,13	5,45	3
4,74	5,00	4,25	4,77	4,88	4,01	4,41	4,92	4,81	4,51	4,58	4,45	4,12	4,49	4,13	4,25	4,42	4,35	3,30	5
3,79	3,80	4,03	3,86	3,72	4,02	3,87	3,87	3,79	3,91	4,03	3,94	4,17	3,82	3,98	3,75	3,72	3,75	4,57	5
4,41	4,29	4,27	4,20	4,30	4,19	4,32	4,29	4,40	4,06	4,11	4,35	4,09	4,25	3,96	4,33	4,35	4,10	5,68	5
3,37	3,51	3,35	3,44	3,44	3,77	3,74	3,55	3,36	3,23	3,25	3,78	3,46	2,94	3,39	3,54	3,46	3,36	5,02	4
3,48	3,70	3,16	3,37	3,52	3,89	3,62	3,62	3,41	3,27	3,33	3,89	2,90	3,18	3,49	3,60	3,34	3,46	5,56	4
3,12	3,20	2,81	2,98	3,26	3,04	2,99	3,22	3,24	2,56	2,60	3,04	2,81	2,49	2,53	2,39	2,56	2,83	7,07	3
2,98	3,18	2,82	2,84	3,41	3,31	3,35	3,15	3,45	3,25	3,34	3,54	2,52	3,23	3,60	3,03	3,21	3,34	5,33	4
3,79	4,97	4,12	3,37	4,75	3,96	4,89	5,00	4,69	4,84	4,86	4,61	4,15	4,62	4,73	4,69	4,79	4,82	3,14	1
3,65	4,24	3,75	3,71	3,85	3,95	4,14	4,29	3,74	3,69	3,66	4,07	3,65	3,95	3,66	4,22	3,10	3,59	5,78	4
4,01	3,55	3,47	3,20	4,00	3,36	3,42	3,55	3,95	3,67	3,60	3,35	3,92	4,15	3,57	4,22	3,67	3,76	4,80	4
4,01	4,22	4,15	4,05	4,57	4,03	4,18	4,12	4,45	4,47	4,35	4,01	4,07	4,31	4,60	4,77	4,38	4,59	3,33	5

.....
... 700k rows

Linear Regression

- Model: $\Omega(\mathbf{x}) = \mathbf{x}^T \mathbf{w}$
- Training: $\mathbf{w} = (\mathbf{X}^T \mathbf{X} + \lambda \mathbf{I})^{-1} \mathbf{X}^T \mathbf{y}$
- ☺ Fast

Determined by cross-validation
 $\lambda = 0.000004$

■ RMSE_{pTest} : **0.87525**
 → **Baseline**

regression
 coefficients
 w_i

-0.083	AFM-1 (0.9362)
-0.084	AFM-2 (0.9231)
-0.077	AFM-3 (0.9340)
+0.088	AFM-4 (0.9391)
+0.098	GE-1 (0.9079)
-0.003	GE-2 (0.9710)
-0.081	GE-3 (0.9443)
+0.176	GE-4 (0.9209)
+0.029	KNN-1 (0.9110)
+0.272	KNN-2 (0.8904)
-0.094	KNN-3 (0.8970)
+0.010	KNN-4 (0.9463)
+0.025	RBM-1 (0.9493)
+0.066	RBM-2 (0.9123)
-0.008	SVD-1 (0.9074)
+0.094	SVD-2 (0.9172)
+0.080	SVD-3 (0.9033)
+0.227	SVD-4 (0.8871)
-0.008	log(support)
+3.673	const. 1

Binned Linear Regression

Lin.Reg Baseline: 0.8752

- Model: $\Omega(\mathbf{x}) = \mathbf{x}^T \mathbf{w}_b$ $b \dots$ bin, each \mathbf{w}_b per bin
- ☺ Fast, more accurate than LR
- 3 binning types
 - support: Number of ratings per user
 - date: Day of the rating
 - frequency: Number of votes from user u on day of the rating

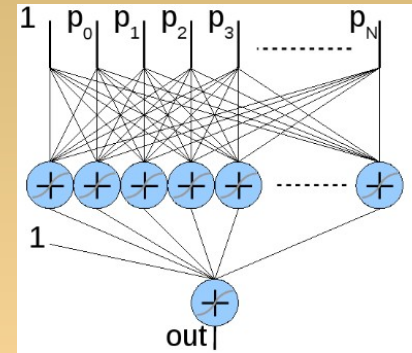
type	2 bins	5 bins	10 bins	20 bins
support	0.874877 (V:0.87517)	0.874741 (V:0.8750)	0.874744 (V:0.87499)	0.87485 (V:0.87513)
date	0.875212 (V:0.87545)	0.875195 (V:0.87541)	0.87527 (V:0.87544)	0.87537 (V:0.87558)
frequency	0.87518 (V:0.87537)	0.87510 (V:0.87521)	0.87512 (V:0.8752)	0.87517 (V:0.87531)

→ support binning works best (5 bins)

Neural Network

- Stochastic gradient descent
- Decrease initial learning rate
- Bagging improves the accuracy
- 😊 Fast and accurate predictions
- ☹ Long training time

Lin.Reg Baseline: 0.8752

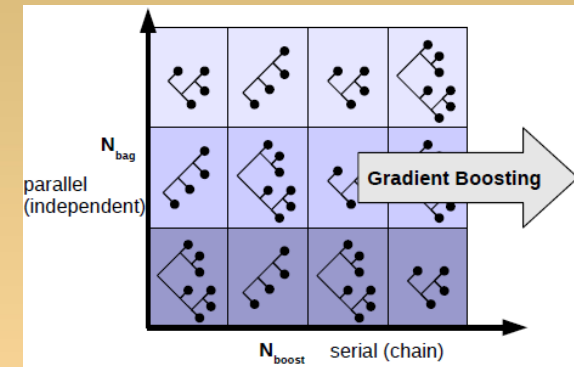


network setup	validation type	RMSE validation	train time	RMSE pTest
19-30-1	retraining 8-CV	0.873633	1.5[h]	0.873365
19-30-1	cross valid. mean 8-CV	0.873633	0.88[h]	0.873316
19-30-1	bagging size=32	0.87347	4.3[h]	0.873191
19-30-1	bagging size=128	0.873436	17.3[h]	0.873185
19-70-1	bagging size=128	0.87342	33.6[h]	0.873163
19-150-1	bagging size=128	0.873473	65.8[h]	0.873169
19-50-30-1	bagging size=128	0.873455	48.6[h]	0.87318

Bagged Gradient Boosted Decision Tree

Lin.Reg Baseline: 0.8752

- Prediction is generated by
 - Splits in a single tree are greedy (best RMSE)
 - Sum of trees (gradient boosting)
 - Averaging many chains (bagging)
- Lower RMSE when
 - Smaller learnrate
 - Larger bagging size
- Dataset dependent
 - Max. Number of leaves
 - Subspace size

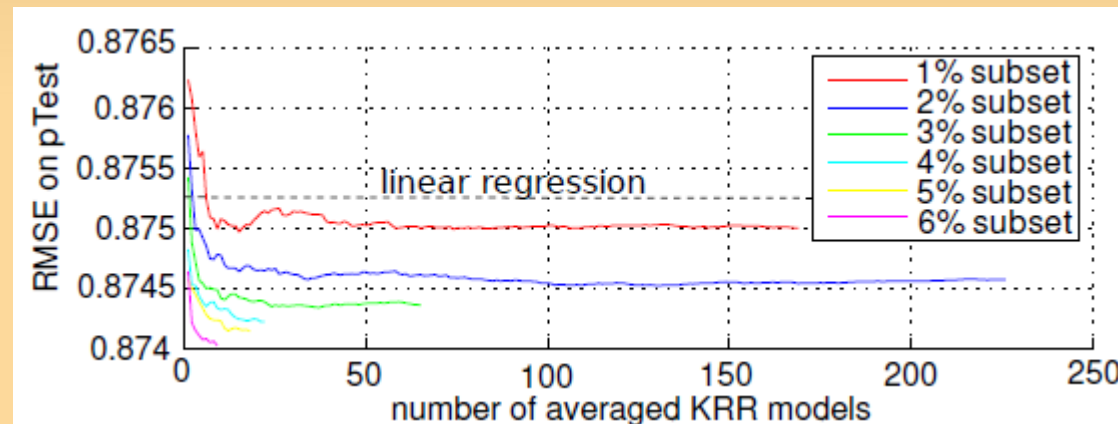


fixed:	$\eta = 0.1$	$\eta = 0.05$	$\eta = 0.03$	$\eta = 0.02$
$N_{bag} = 32$	0.874783	0.87467	0.874624	0.874593
$K = 300$	0.87437	0.874352	0.87433	0.874309
$S = 2$	0.62[h]	1.21[h]	1.94[h]	3.27[h]
fixed:	$K = 500$	$K = 300$	$K = 200$	$K = 100$
$N_{bag} = 32$	0.874838	0.874783	0.874767	0.874934
$\eta = 0.1$	0.874427	0.87437	0.874399	0.874546
$S = 2$	0.48[h]	0.62[h]	0.73[h]	1.39[h]
fixed:	$N_{bag} = 16$	$N_{bag} = 32$	$N_{bag} = 64$	$N_{bag} = 128$
$\eta = 0.02$	0.874936	0.874593	0.874554	0.874517
$K = 300$	0.874381	0.874309	0.874293	0.874288
$S = 2$	1.1[h]	3.27[h]	7.01[h]	14.66[h]
fixed:	$S = 1$	$S = 2$	$S = 4$	$S = 8$
$\eta = 0.1$	0.874838	0.874784	0.87477	0.874841
$K = 500$	0.874427	0.874377	0.874405	0.874504
$N_{bag} = 32$	0.48[h]	0.42[h]	0.68[h]	1.11[h]

Kernel Ridge Regression

Lin.Reg Baseline: 0.8752

- Cannot be applied to all 700k training samples
 - $O(N^3)$ runtime, $O(N^2)$ memory
- Average over smaller trainsets (random x % subset)



RMSE: 0.874

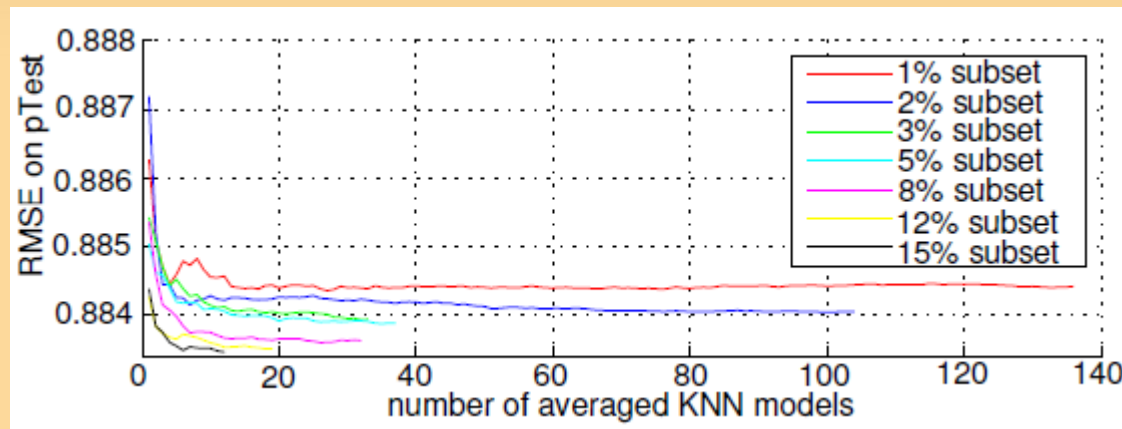
1% subset: 7k samples

6% subset: 42k samples

K-Nearest Neighbors Blending

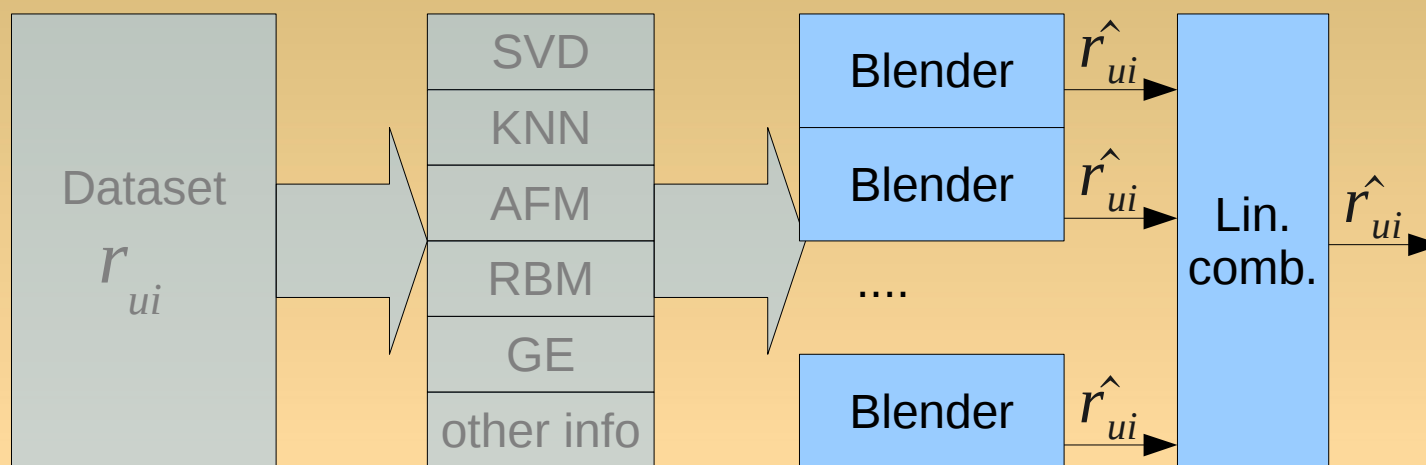
Lin.Reg Baseline: 0.8752

- Cannot be applied to all 700k training samples
 - $O(N^2)$ runtime, $O(N^2)$ memory
- Does not work (worse RMSE)



RMSE: 0.883

Bagging with Neural Networks, Polynomial Regression and GBDT



Many Blenders are trained one after another

- Error feedback for stop training:
RMSE of the linear combination
- Linear Combination is calculated on the
out-of-bag estimate

Bagging with Neural Networks, Polynomial Regression and GBDT

Lin.Reg Baseline: 0.8752

- Stagewise optimization of a lin. combination of different learners

model	RMSE (blend)	weight	parameters
const. 1	-	-0.014	-
NN	0.87345 (0.873445)	0.170	19-100-1, $\alpha = 3.6$, $\beta = 3.0$, $\eta = 5e-4$, $\eta^{(-)} = 5e-7$, 870 epochs, 44.4[h]
GBDT	0.874111 (0.873387)	0.054	$S = 20$, $K = 50$, $\eta = 0.1$, 226 epochs, randomSplitPoint, 6.6[h]
GBDT	0.874603 (0.873384)	0.098	$S = 2$, $K = 300$, $\eta = 0.02$, 267 epochs, optSplitPoint, 8.1[h]
PR	0.874358 (0.87336)	0.141	order=2, $\lambda = 2.4e-6$, with cross interactions, 1.9[h]
PR	0.895951 (0.873351)	-0.033	order=3, $\lambda = 0.054$, no cross interactions, 0.3[h]
NN	0.87345 (0.873296)	0.202	19-100-1, $\alpha = 2$, $\beta = 3.0$, $\eta = 5e-4$, $\eta^{(-)} = 5e-7$, 998 epochs, 47.1[h]
NN	0.873449 (0.873227)	0.371	19-50-30-1, $\alpha = 2$, $\beta = 3.0$, $\eta = 5e-4$, $\eta^{(-)} = 5e-7$, 952 epochs, 49.8[h]
blend	0.873227 pTest:0.87297		total train time: 158.2[h] total prediction time 4.5[h]

Results on qualifying set (the "real" test set)

Lin.Reg Baseline: 0.8681

- 19-30-1 neural network: RMSE=0.8664
- Bagging with 7 models: RMSE=**0.8660**

0.0021 improvement

- Netflix Prize competitors use linear regression with meta features

Meta-Feature	Probe CV RMSE	Test RMSE
1	0.869889	0.863377
...
25	0.867501	0.861405

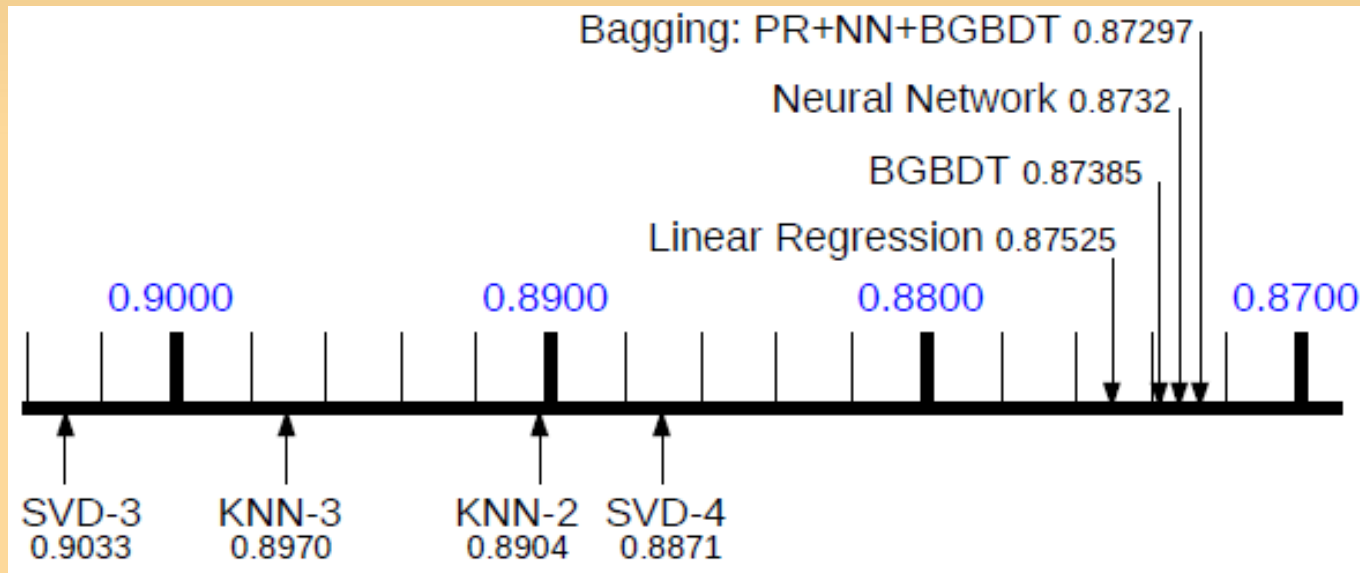
0.0020 improvement

Table 2: RMSEs Using Cumulative Meta-Feature Sets

[J. Sill, G. Takacs, L. Mackey, and D. Lin. : **Feature-weighted linear stacking**, 2009]

Summary

- The blend of many CF algorithms improves the accuracy!
- Neural network (as blender) is the best tradeoff between training time and accuracy



blended algorithms

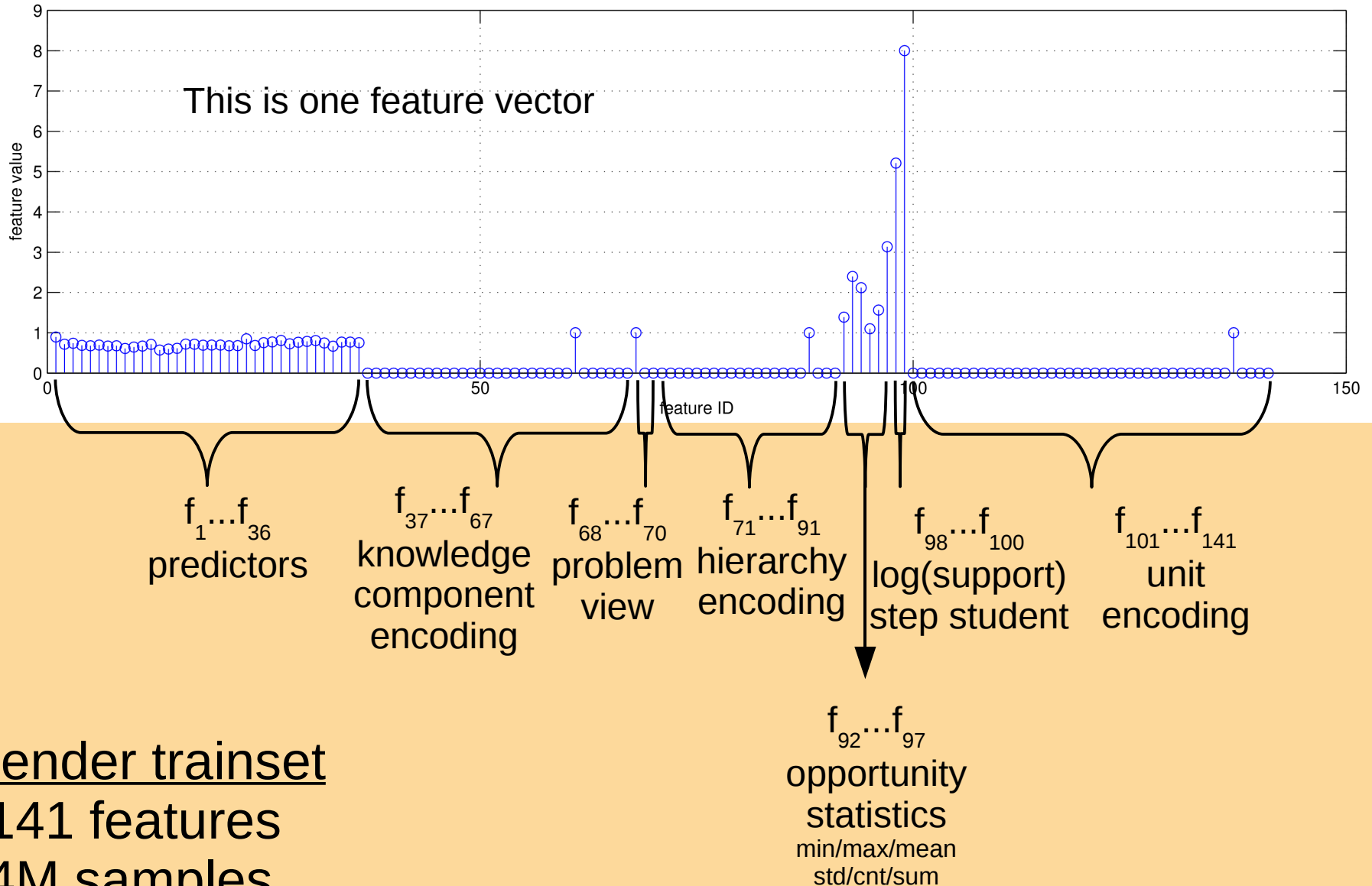
individual collaborative filtering algorithms

Software is Open Source!

- The data and the implementation can be found on:
<http://elf-project.sourceforge.net/>
- Many examples are provided there

Happy hacking 😊

Application example: KDD Cup 2010



Blender trainset

- 141 features
- 4M samples

Thank you for your attention!

Michael Jahrer
commendo research & consulting GmbH
www.commendo.at