Journal of Machine Learning Research (2010) Submitted ; Published

Collaborative Filtering Applied to Educational Data Mining

Andreas Töscher ANDREAS.TOESCHER@COMMENDO.AT
commendo research
8580 Köflach, Austria

Michael Jahrer MICHAEL.JAHRER@COMMENDO.AT
commendo research
8580 Köflach, Austria

Editor:

Abstract

We present our overall third ranking solution for the KDD Cup 2010 on educational data mining. The goal of the competition was to predict a student's ability to answer questions correctly, based on historic results. In our approach we use an ensemble of collaborative filtering techniques, as used in the field of recommender systems and adopt them to fit the needs of the competition. The ensemble of predictions is finally blended, using a neural network.

Keywords: Collaborative Filtering, KDD Cup, Educational Data Mining

1. Introduction

The goal of the KDD Cup 2010 on educational data mining is to predict the correctness of a student's first attempt to solve a problem. The cup organizers provide 2 previously unpublished datasets, called *Algebra 2008-2009* and *Bridge to Algebra 2008-2009*. These two datasets contain the same data fields but vary in size.

This years KDD Cup has very interesting similarities to the field of recommender systems. In the world of recommender systems there are users and items, and interactions between them, like ratings and purchases. So one ends up with a big user-item matrix with lots of elements missing. The goal is to predict the missing values. In the KDD Cup 2010 there are students and steps. Not every student has tried to answer every step, so the resulting matrix is also sparse. Thus the basic problem looks very similar, for this reason we adopted methods from the field of recommender systems to fit the needs of this cup.

1.1 Data and Notation

The goal was to predict whether a student solves a step at the first attempt. Throughout the text we denote the step with i and the student with s.

Both datasets provided have the same structure, but vary in size. The smaller *Algebra 2008-2009* has nearly 9 million lines, while the bigger *Bridge to Algebra 2008-2009* has over 20 million. Exact numbers can be found in Table 1. The datasets are generated from students answering questions from an electronic tutoring system. We denote the set of students with $\mathbb{S}$. Each student has to answer steps, the set of steps is denoted as $\mathbb{I}$. In case

	Algebra 2008-2009	Bridge to Algebra 2008-2009
Lines (train)	8,918,054	20,012,498
Students (train)	3,310	6,043
Steps (train)	1,357,180	603,176
Problem (train)	211,529	63,200
Section (train)	165	186
Units (train)	42	50
KC (train)	2,097	1,699
Steps (new on test)	4,390	9,807

Table 1: The table above contains general dataset statistics. In the last line we see, that both datasets contain steps in the test set, which are not present in the train set. These cases are a challenge for collaborative filtering.

a student $s \in \mathbb{S}$ answers the step $i \in \mathbb{I}$ correctly in the first attempt c_{is} is 1 otherwise 0. Not every student answers every step, so the matrix $\mathbf{C} = [c_{is}]$ is sparse. The set of students who answered step i is called $\mathbb{S}_i$, hence the set of steps answered by student s is $\mathbb{I}_s$.

Steps are grouped into problems, so every step belongs to exactly one problem. The set of all problems is denoted as $\mathbb{P}$ and $p(i)$ is the problem of step i. Furthermore, problems are grouped into sections, and sections into units. We denote the set of all sections as $\mathbb{X}$ and the section of step i as $x(i)$. The set of units is $\mathbb{U}$ and the unit of step i is $u(i)$. Every step has set of knowledge components (KCs), which describe the skills needed for solving the step. Due to the fact that the set of KCs associated with a step changes over time, we denote the set of KCs for a given student step pair as $\mathbb{K}(i, s)$.

For the competition the set $\mathbb{L} = \{(i, s)|i \in \mathbb{I}, s \in \mathbb{S}, c_{is}\ is\ known\}$ consisting of known elements of the matrix $\mathbf{C}$ is divided into a training set $\mathbb{L}_T$ and a test set $\mathbb{L}_P$, so that $\mathbb{L}_T \cup \mathbb{L}_P = \mathbb{L}$ and $\mathbb{L} \cap \mathbb{T} = \emptyset$. The error measure is the root mean square error (RMSE). The test set RMSE is given by $\sqrt{\frac{1}{|\mathbb{L}_T|} \sum_{(i,s)\in\mathbb{L}_T} (c_{is} - \hat{c}_{is})^2}$, while $\hat{c}_{is}$ is the predicted value of student s having step i correct at the first time. The leaderboard scores are calculated based on an unknown subset of the test set $\mathbb{L}_T$.

1.2 Framework

During the few weeks of the KDD cup we tried many methods and ways to model the data. Every model was trained and evaluated using a 8-fold cross validation. Finally we used a neural network to blend the collected ensemble of predictions from different models.

For the cross validation we divide the set $\mathbb{L}_T$ into 8 equally sized disjoint sets $\mathbb{L}_{T_1}$ to $\mathbb{L}_{T_8}$. So we train 8 models in parallel, the first trains on the set $\mathbb{L}_{T_2} \cup \mathbb{L}_{T_3} \cup ... \cup \mathbb{L}_{T_8}$ and generates predictions for the set $\mathbb{L}_{T_1}$ and the test set $\mathbb{L}_P$. The second trains on the set $\mathbb{L}_{T_1} \cup \mathbb{L}_{T_3} \cup \mathbb{L}_{T_4} \cup ... \cup \mathbb{L}_{T_8}$ and predicts for set $\mathbb{L}_{T_2}$ and the test set $\mathbb{L}_P$. So every model predicts one part of the train set and the test set $\mathbb{L}_P$, while being trained on the remaining data. Hence we have 8 predictions for every c_{is} in the test set. For the test set we use the mean of the 8 probe predictions. For this reason our probe RMSE values are lower than

the RMSE values calculated by cross validation. Throughout the paper all reported RMSE values are the values stemming from cross validation.

2. Algorithms

As stated in the introductory section we adopted the ideas of collaborative filtering to fit the needs of KDD Cup 2010. For every algorithm we describe the origin, and our modifications and report all results used in our final submission.

2.1 K Nearest Neighbor (KNN)

One of the most naïve algorithms is to use the nearest neighbors for prediction. In the world of recommender systems this algorithm is known as user based collaborative filtering. It was originally published by Resnick et al. (1994). For the Netflix competition the authors used a modified version with correlation shrinkage, which can be found in Töscher and Jahrer (2008).

The idea of this algorithm is to search for the K students who have the most similar results in history and use a weighted mean of their results for prediction. For calculating similarities between users we use the Pearson correlation between students calculated on the subset of commonly answered steps.

The steps answered by student s_1 and s_2 are given by $\mathbb{I}_{s_1}$ and $\mathbb{I}_{s_2}$. Based on the common subset of items $\mathbb{I}_{s_1 s_2} = \mathbb{I}_{s_1} \cap \mathbb{I}_{s_2}$ answered by both students the Pearson correlation between them is given by

$$\rho_{s_1 s_2} = \frac{\frac{1}{|\mathbb{I}_{s_1 s_2}|-1} \sum_{i \in \mathbb{I}_{s_1 s_2}} (c_{s_1 i} - \mu_{s_1})(c_{s_2 i} - \mu_{s_2})}{\sqrt{\frac{1}{|\mathbb{I}_{s_1 s_2}|-1} \sum_{i \in \mathbb{I}_{s_1 s_2}} (c_{s_1 i} - \mu_{s_1})^2} \sqrt{\frac{1}{|\mathbb{I}_{s_1 s_2}|-1} \sum_{i \in \mathbb{I}_{s_1 s_2}} (c_{s_2 i} - \mu_{s_2})^2}}, \tag{1}$$

where

$$\mu_{s_1} = \frac{1}{|\mathbb{I}_{s_1 s_2}|} \sum_{i \in \mathbb{I}_{s_1 s_2}} c_{s_1 i} \tag{2}$$

and

$$\mu_{s_2} = \frac{1}{|\mathbb{I}_{s_1 s_2}|} \sum_{i \in \mathbb{I}_{s_1 s_2}} c_{s_2 i} \tag{3}$$

being the mean values. The cardinality $|\mathbb{I}_{s_1 s_2}|$ of the set of commonly answered steps varies between different students. Therefore we shrink the correlation

$$\bar{\rho}_{s_1 s_2} = \frac{|\mathbb{I}_{s_1 s_2}| \cdot \rho_{s_1 s_2}}{|\mathbb{I}_{s_1 s_2}| + \alpha} \tag{4}$$

towards zero based on the relation between the number of samples used to calculate the correlation and the meta parameter α. According to our experience the shrinkage is very important because it accounts for the strongly varying number of samples used to calculate the correlation. After the shrinkage we apply a nonlinear mapping using the sigmoid function $\sigma(x) = \frac{1}{1-e^{-x}}$. This leads to

$$\tilde{\rho}_{s_1 s_2} = \sigma\left(\delta \cdot \bar{\rho}_{s_1 s_2} + \gamma\right), \tag{5}$$

Dataset	RMSE	Meta parameters
Algebra 2008-2009	0.3257	$K = 41, \alpha = 12.9, \beta = 1.5, \delta = 6.2, \gamma = -1.9$
Bridge to Algebra 2008-2009	0.3049	$K = 41, \alpha = 12.9, \beta = 1.5, \delta = 6.2, \gamma = -1.9$

Table 2: This table reports all results optained by the student based KNN described in Section 2.1 and used for the final submission.

where the meta parameters δ and γ control the mapping. In order to give predictions for student s on step i we use a weighted average of the K most similar students who answered step i. This is given by

$$\tilde{c}_{is} = \frac{\sum_{\tilde{s} \in \mathbb{S}_i(s;K)} \tilde{\rho}_{s\tilde{s}} c_{i\tilde{s}}}{\sum_{\tilde{s} \in \mathbb{S}_i(s;K)} |\tilde{\rho}_{s\tilde{s}}|} \tag{6}$$

where $\mathbb{S}_i(s;K) \subset \mathbb{S}_i$ is a set of K students with the highest correlation to student s. In some cases there are few or no other students who answered the step i or they have only weak correlations to student s. For this reason we correct the predicted $\tilde{c}_{is}$ towards the student mean μ_s

$$\hat{c}_{is} = \frac{\tilde{c}_{is} \sum_{\tilde{s} \in \mathbb{S}_i(s;K)} |\tilde{\rho}_{s\tilde{s}}| + \mu_s \beta}{\sum_{\tilde{s} \in \mathbb{S}_i(s;K)} |\tilde{\rho}_{s\tilde{s}}| + \beta} \tag{7}$$

based on the relation between the sum of correlations and the meta parameter β. If the sum of correlations is big compared to β, the prediction $\tilde{c}_{is}$ is influenced marginally, but if the sum of correlations is low, the prediction is strongly corrected towards the user mean. In the extreme case the prediction is the user mean. This is very important for steps which have not been answered by other students.

In Table 2 we report results of this algorithm for both datasets.

2.2 Singular Value Decomposition (SVD)

The singular value decomposition is well known in linear algebra and states that an arbitrary matrix $\mathbf{C}$ with real or complex entries can be decomposed into a product of three matrices $\mathbf{A\Sigma B}^T$ with $\mathbf{\Sigma}$ being a diagonal matrix. In the collaborative filtering context the matrix $\mathbf{C}$ is sparse, meaning that most of the elements are unknown. The basic idea of SVD to decompose the matrix $\mathbf{C}$ can be still applied. During the Netflix competition, this method became very popular because of two reasons. First it delivers good results out of the box and second it is easy to tune and customize. Over the last years there were lots of publications describing extensions to the basic SVD. We first show how to use the basic SVD and then how to extend it.

The idea is to represent a step i by a feature vector $\mathbf{a}_i$ and a student s by a feature vector $\mathbf{b}_s$. This means the steps are represented by a $|\mathbb{I}| \times N$ matrix $\mathbf{A}$, where N is the number of features used to represent a step. The students are represented by a $|\mathbb{S}| \times N$ matrix $\mathbf{B}$. The final goal is that the product of the matrices $\mathbf{AB}^T$ approximates the known elements of $\mathbf{C}$. This leads to $min_{\mathbf{A},\mathbf{B}} \|\mathbf{C} - \mathbf{AB}^T\|_C^2$, where the norm $\|\cdot\|_C^2 = \sum_{(i,s) \in \mathbb{L}_T} c_{is}^2$ is only defined on the known elements of $\mathbf{C}$. In order to account for the different number

Dataset	RMSE	Meta parameters
Algebra 2008-2009	0.446277	$N = 10, \eta = 0.002, \lambda = 0.02$
Bridge to Algebra 2008-2009	0.3168	$N = 10, \eta = 0.002, \lambda = 0.02$
Bridge to Algebra 2008-2009	0.3159	$N = 20, \eta = 0.002, \lambda = 0.01$
Bridge to Algebra 2008-2009	0.3178	$N = 20, \eta = 0.002, \lambda = 0.03$

Table 3: This table reports all results obtained by the SVD in Section 2.2. The parameter N is the feature size, η is the learn rate and λ the L2 regularization. The results on the *Bridge to Algebra* dataset look promising, while the results on the *Algebra* dataset do not. The reason for the bad results on the first dataset is the large number of steps with few answers, which leads to few updates of the corresponding feature vectors. The introduction of biases in the improved version in Section 2.3 corrects this problem.

of known elements for students and steps, regularizing the matrix decomposition is very important. This leads to the following error function to be minimized:

$$E(\mathbf{A}, \mathbf{B}) = \|\mathbf{C} - \mathbf{A}\mathbf{B}^T\|_C^2 + \lambda(\|\mathbf{A}\|_F^2 + \|\mathbf{B}\|_F^2) \tag{8}$$

In the equation above $\|\cdot\|_F^2$ denotes the squared Frobenius norm and λ is the meta parameter for controlling the L2 regularization. A rating prediction is then given by

$$\hat{c}_{is} = \mathbf{a}_i^T \cdot \mathbf{b}_s \tag{9}$$

a dot product of the corresponding student and step feature vectors. Using the above prediction leads to the following error function:

$$E = \sum_{(i,s)\in\mathbb{L}_T} (c_{is} - \hat{c}_{is})^2 + \lambda(\|\mathbf{A}\|_F^2 + \|\mathbf{B}\|_F^2) \tag{10}$$

In order to train this model we initialize all parameters from a uniform distribution $[-0.001, 0.001]$ and train the model using stochastic gradient descent. In Table 3 we report results on both datasets.

2.3 Factor Model 1 (FM1)

The FM1 enhances the idea of the SVD model, and corrects some shortcomings of pure SVD, as described in Section 2.2. The extensions are inspired by the work of Koren (2010) and Paterek (2007). We saw that failing to use biases generates large errors for unknown students or steps. Hence this model uses lots of biases. We have the global bias μ, a step dependent bias $\hat{\mu}_i$ and a student dependent bias $\bar{\mu}_s$. Additionally, we include a unit dependent bias $\breve{\mu}_{u(i)}$, a section bias $\acute{\mu}_{x(i)}$ and a problem bias $\tilde{\mu}_{p(i)}$. We found that the knowledge component (KC) includes valuable information, unfortunately the number of KCs is different for every step and also changes over time. So we denote student and step dependent set of KCs as $\mathbb{K}(i, s)$, and introduce a KC dependent bias $\check{\mu}_k$ and a student and

Dataset	RMSE	Meta parameters
Algebra 2008-2009	0.3078	$N = 50, \eta = 0.0005, \lambda = 0.01$
Bridge to Algebra 2008-2009	0.3013	$N = 50, \eta = 0.0005, \lambda = 0.01$

Table 4: This table reports all results obtained by FM1 in Section 2.3. The parameter N is the feature size, η is the learning rate and λ the L2 regularization.

KC dependent bias $\dot{\mu}_{ks}$. We also use KC dependent N dimensional feature vectors $\acute{\mathbf{a}}_k$, which are added to the step feature vector $\mathbf{a}_i$. The introduced biases and feature vectors enhance the basic SVD prediction from Equation 9 to

$$\hat{c}_{is} = \mu + \hat{\mu}_i + \bar{\mu}_s + \tilde{\mu}_{p(i)} + \acute{\mu}_{x(i)} + \breve{\mu}_{u(i)} + \frac{1}{\sqrt{|\mathbb{K}(i,s)|}} \sum_{k \in \mathbb{K}(i,s)} (\breve{\mu}_k + \dot{\mu}_{ks}) \tag{11}$$

$$+ \left(\mathbf{a}_i + \frac{1}{\sqrt{|\mathbb{K}(i,s)|}} \sum_{k \in \mathbb{K}(i,s)} \acute{\mathbf{a}}_k \right)^T \cdot \mathbf{b}_s \tag{12}$$

the prediction equation for the FM1 model.

Like in the SVD case we use stochastic gradient descent with L2 regularization to train all parameters. The obtained results using this model are shown in Table 4. The introduction of the biases and the KC dependent feature vector significantly lowered the RMSE on both datasets.

2.4 Factor Model 2 (FM2)

The FM2 is very similar to the FM1. We remove the student and KC dependent bias $\dot{\mu}_{ks}$ and introduced a student and unit dependent bias $\ddot{\mu}_{su(i)}$. This leads to the following prediction equation:

$$\hat{c}_{is} = \mu + \hat{\mu}_i + \bar{\mu}_s + \tilde{\mu}_{p(i)} + \breve{\mu}_{u(i)} + \ddot{\mu}_{su(i)} + \frac{1}{\sqrt{|\mathbb{K}(i,s)|}} \sum_{k \in \mathbb{K}(i,s)} \breve{\mu}_k \tag{13}$$

$$+ \left(\mathbf{a}_i + \frac{1}{\sqrt{|\mathbb{K}(i,s)|}} \sum_{k \in \mathbb{K}(i,s)} \acute{\mathbf{a}}_k \right)^T \cdot \mathbf{b}_s \tag{14}$$

We train the model using stochastic gradient descent with L2 regularization. The obtained results are shown in Table 5.

2.5 Factor Model 3 (FM3)

This algorithm extends the idea of the first two factor models by modeling even more relationships in the data. A global bias term is represented by

$$\breve{\mu}_{is} = \mu + \hat{\mu}_i + \bar{\mu}_s + \tilde{\mu}_{p(i)} + \acute{\mu}_{x(i)} + \breve{\mu}_{u(i)} + \frac{1}{\sqrt{|\mathbb{K}(i,s)|}} \sum_{k \in \mathbb{K}(i,s)} \breve{\mu}_k, \tag{15}$$

Dataset	RMSE	Meta parameters
Algebra 2008-2009	0.3044	$N = 150, \eta = 0.002, \lambda = 0.005$
Algebra 2008-2009	0.3045	$N = 150, \eta = 0.003, \lambda = 0.003$
Bridge to Algebra 2008-2009	0.2997	$N = 100, \eta = 0.003, \lambda = 0.005$

Table 5: The reported results stem from the FM2. The higher feature size N and the higher regularization λ combined with the small modeling changes, significantly improved the RMSE compared to the FM1 model.

Dataset	RMSE	Meta parameters
Algebra 2008-2009	0.2996	$N = 5$
Bridge to Algebra 2008-2009	0.2916	$N = 20$
Bridge to Algebra 2008-2009	0.2924	$N = 5$

Table 6: This table contains our best results obtained with FM3.

where μ is the global, $\hat{\mu}_i$ the step and $\bar{\mu}_s$ the student bias. A problem bias is modeled by $\tilde{\mu}_{p(i)}$, a section bias by $\acute{\mu}_{x(i)}$ and a unit bias is represented as $\breve{\mu}_{u(i)}$. As in FM2 we model a knowledge component specific bias $\breve{\mu}_k$, but we removed the user specific knowledge component bias.

This model extends the N dimensional step feature vector $\mathbf{a}_i$ for step i, by adding a unit dependent feature vector $\hat{\mathbf{a}}_{u(i)}$, a section feature vector $\bar{\mathbf{a}}_{x(i)}$ and a problem feature vector $\tilde{\mathbf{a}}_{p(i)}$. We further introduce knowledge component dependent feature vectors $\acute{\mathbf{a}}_k$, which leads to:

$$\breve{\mathbf{a}}_{is} = \mathbf{a}_i + \hat{\mathbf{a}}_{u(i)} + \bar{\mathbf{a}}_{x(i)} + \tilde{\mathbf{a}}_{p(i)} + \frac{1}{\sqrt{|\mathbb{K}(i,s)|}} \sum_{k \in \mathbb{K}(i,s)} \acute{\mathbf{a}}_k \tag{16}$$

The students are modeled as

$$\breve{\mathbf{b}}_{is} = \mathbf{b}_s + \hat{\mathbf{b}}_{u(i)} + \bar{\mathbf{b}}_{x(i)} + \tilde{\mathbf{b}}_{p(i)} + \frac{1}{\sqrt{|\mathbb{K}(i,s)|}} \sum_{k \in \mathbb{K}(i,s)} \acute{\mathbf{b}}_k, \tag{17}$$

where $\mathbf{b}_s \in \mathbb{R}^N$ is a N dimensional student feature vector. Similar to the step side we use feature vectors per unit $\hat{\mathbf{b}}_{u(i)}$, per section $\bar{\mathbf{b}}_{x(i)}$ and per problem $\tilde{\mathbf{b}}_{p(i)}$. We use knowledge component dependent features $\acute{\mathbf{b}}_k$.

A prediction is given by:

$$\hat{c}_{is} = \breve{\mu}_{is} + \breve{\mathbf{a}}_{is}^T \cdot \breve{\mathbf{b}}_{is} \tag{18}$$

In order to train this model we use stochastic gradient descent to minimize a quadratic error function. In contrast to the former models we do not use the same learning rate and L2 regularization for every parameter. We use different learning rates and regularizations for every bias and feature vector. Our best results obtained are reported in Table 6.

2.6 Group Factor Model (GFM)

One of our biggest concerns during the few weeks of the competition was to miss an important signal in the data. This was the main driving force behind this model. We try to include every information available and model all possible interactions.

First of all we have to introduce additional notation. So far we have used the fact that a step belongs to one problem, one section and one unit. In FM1 to FM3 we already use the knowledge components. For this model we use the knowledge component in a different way, we do not parse the provided string into the individual knowledge components. We directly map the string to an ID. So every KC string which occurs more than 5 times gets its own ID, all the others share a default ID. Furthermore, we use the opportunity count which was provided for each KC. We directly use the string for grouping. Similar to the KC string, we only use opportunity count group strings which occur more than 5 times, all the others share a default group. An additional grouping is given by the name of the steps. Some steps have the same name but belong to different problems, so we introduce a grouping based on the plain step name. The grouping we use is based on the problem view, we have one group for the problem view being 1 and one group for the rest. Hence we end up having 7 different groupings.

A prediction $\hat{c}_{is}$ for student s on step i is given by:

$$\hat{c}_{is} = \mu + \sum_{g=1}^{7} \mathbf{b}_s^T \cdot \mathbf{d}_{g,G^{(g)}(i,s)} + \sum_{g=1}^{7} \bar{\mathbf{a}}_i^T \cdot \bar{\mathbf{d}}_{g,G^{(g)}(i,s)} \tag{19}$$

$$+ \sum_{g=1}^{7} \tilde{\mathbf{b}}_{sg}^T \cdot \tilde{\mathbf{d}}_{g,G^{(g)}(i,s)} + \sum_{g=1}^{7} \hat{\mathbf{a}}_{ig}^T \cdot \hat{\mathbf{d}}_{g,G^{(g)}(i,s)} \tag{20}$$

$$+ \sum_{g=1}^{7} \sum_{\hat{g}=1}^{7} \breve{\mathbf{d}}_{g,G^{(g)}(i,s)}^T \cdot \acute{\mathbf{d}}_{\hat{g},G^{(\hat{g})}(i,s)} \tag{21}$$

$$+ \sum_{g=1}^{7} \sum_{\hat{g}=g+1}^{7} \breve{\mathbf{d}}_{g,G^{(g)}(i,s)}^T \cdot \breve{\mathbf{d}}_{\hat{g},G^{(\hat{g})}(i,s)} \tag{22}$$

Every student s and step i pair belongs to exactly 7 groups, while the group ID for group g is given by $G^{(g)}(i,s)$. All feature vectors are N dimensional.

We train the model using stochastic gradient descent. In Table 7 we report our best results using this model.

2.7 Restricted Boltzmann Machines (RBM)

RBMs, as described in Salakhutdinov et al. (2007), got very popular for collaborative filtering during the Netflix competition. They were especially known to achieve great results in a big ensemble with SVD and factor models. We tried to use RBMs on the KDD Cup dataset and the results looked very promising, unfortunately we were not able to finalize this model before the deadline.

Dataset	RMSE	Meta parameters
Algebra 2008-2009	0.2997	$N = 20, \eta = 0.003, \eta^- = 0.0001, \lambda = 0.001$
Algebra 2008-2009	0.3051	$N = 20, \eta = 0.002, \eta^- = 0.0001, \lambda = 0.0$
Bridge to Algebra 2008-2009	0.2908	$N = 20, \eta = 0.005, \eta^- = 0.0002, \lambda = 0.0016$
Bridge to Algebra 2008-2009	0.2901	$N = 50, \eta = 0.005, \eta^- = 0.0002, \lambda = 0.0016$
Bridge to Algebra 2008-2009	0.2898	$N = 60, \eta = 0.005, \eta^- = 0.0002, \lambda = 0.0016$

Table 7: This table shows results of the GFM. The used feature size is called N. We used stochastic gradient descent with L2 regularization. The learning rate is denoted with η and the regularization with λ. In order to slightly improve the results we decreased the learning rate by a fixed amount η^- after every epoch.

3. Blending

The individual predictions stemming from the methods described in Section 2 are combined using a blender. This is a typical approach to improve the prediction accuracy, which was shown in Jahrer et al. (2010). It is widely known that an ensemble of predictions performs better than the best individual result. The training set for the blender consists of the predictions we get from the cross validation process from each individual collaborative filtering algorithm. As an effect of the ongoing competition we blended more individual results as listed in the previous sections. They stem from parameter tweaks. For the first dataset (Algebra 2008-2009) we use 36 predictors, for the second dataset (Bridge to Algebra 2008-2009) 37 predictors are used. After a few experiments with different learners and different parameters it turned out that a neural network with two hidden layer works best. The net layout for the first dataset is 142-80-80-1 and for the second dataset 130-80-80-1. Both blending networks are trained for 90 epochs, the learning rate $\eta = 0.001$ is subtracted by 10^{-5} every epoch. No weight decay is used. The blending software is taken from the ELF-project [Jahrer (2010)]. The features in the training set, which are now predictions from CF algorithms, are enlarged by additional information. Figure 1 is a visualization of one feature vector of the blending training set from the dataset Algebra 2008-2009. We add knowledge component encoding, problem view encoding, hierarchy encoding, unit encoding, opportunity statistics and the student/step support. Encoding is a binary information source, where exactly one input has value 1.0.

Since both datasets are huge in the number of samples (9 million for *Algebra 2008-2009*, 20 million for *Bridge to Algebra 2008-2009*) the blender needs a remarkable amount of time for training. The blending is done on all samples of the training set. We make experiments with subsampling and surprisingly the RMSE of the submission improves (ca. 0.0020) when we take the last 40% of ratings per student and unit. These reflects the distribution of the test set better. The RMSE values of the blending (cross-validation) and the submission are listed in Table 8.

	Dataset: Algebra 2008-2009	Dataset: Bridge to Algebra 2008-2009
blend	0.280925: on cross-validation	0.288417: on cross-validation
blend	0.277355: submission	0.28073: submission

Table 8: RMSE values from the blend of both datasets.

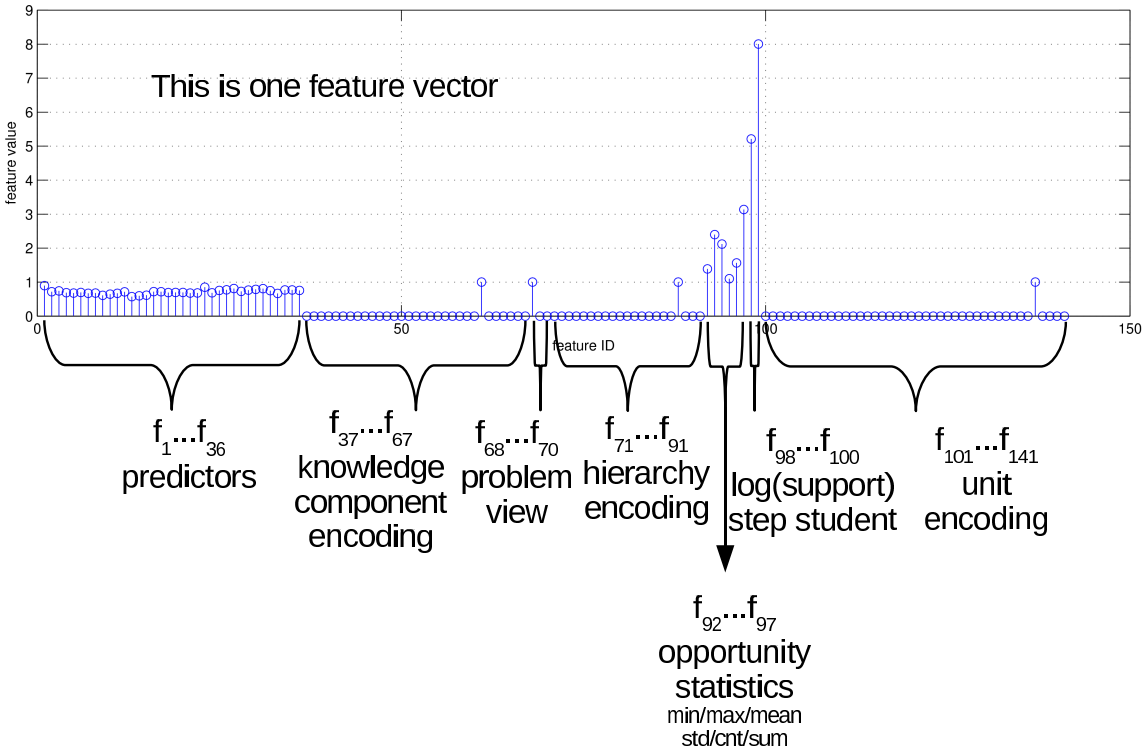

Figure 1: Numerical values of one feature vector of the training set for blending. This example is taken from the *Algebra 2008-2009* dataset.

4. Conclusion

We have shown how to use ideas from collaborative filtering for educational datamining. We adopted KNN and matrix factorization to be applicable to the provided datasets. Finally, we blended an ensemble of predictions using a neural network. The resulting solution ranked third in the KDD Cup 2010.

References

Michael Jahrer. ELF - Ensemble Learning Framework. An open source C++ framework for supervised learning. http://elf-project.sourceforge.net, 2010.

Michael Jahrer, Andreas Töscher, and Robert Legenstein. Combining predictions for accurate recommender systems. In *KDD: Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining*, 2010.

Yehuda Koren. Collaborative filtering with temporal dynamics. *Commun. ACM*, 53(4): 89–97, 2010. ISSN 0001-0782. doi: http://doi.acm.org/10.1145/1721654.1721677.

Arkadiusz Paterek. Improving regularized singular value decomposition for collaborative filtering. *Proceedings of KDD Cup and Workshop*, 2007.

Paul Resnick, Neophytos Iacovou, Mitesh Suchak, Peter Bergstrom, and John Riedl. Grouplens: an open architecture for collaborative filtering of netnews. In *CSCW '94: Proceedings of the 1994 ACM conference on Computer supported cooperative work*, pages 175–186, New York, NY, USA, 1994. ACM. ISBN 0-89791-689-1. doi: http://doi.acm.org/10.1145/192844.192905.

Ruslan Salakhutdinov, Andriy Mnih, and Geoffrey E. Hinton. Restricted boltzmann machines for collaborative filtering. In *ICML*, pages 791–798, 2007.

Andreas Töscher and Michael Jahrer. The BigChaos Solution to the Netflix Prize 2008. Technical report, commendo research & consulting, 2008.