# Solutions and Experiences from KDD Cup 2011

## Track 1: A Linear Ensemble of Individual and Blended Models for Music Rating Prediction

Po-Lung Chen, Chen-Tse Tsai, Yao-Nan Chen, Ku-Chun Chou, Chun-Liang Li, Cheng-Hao Tsai, Kuan-Wei Wu, Yu-Cheng Chou, Chung-Yi Li, Wei-Shih Lin, Shu-Hao Yu, Rong-Bing Chiu, Chieh-Yen Lin, Chien-Chih Wang, Po-Wei Wang, Wei-Lun Su, Chen-Hung Wu, Tsung-Ting Kuo, Todd G. McKenzie, Ya-Hsuan Chang, Chun-Sung Ferng, Chia-Mau Ni, Hsuan-Tien Lin, Chih-Jen Lin and Shou-De Lin

National Taiwan University

# What is KDD Cup?

## Background

- an annual competition on KDD (knowledge discovery and data mining)
- organized by ACM SIGKDD, starting from 1997, now **the most prestigious data mining competition**
- usually lasts 3-4 months
- participants include famous research labs (IBM, AT&T) and top universities (Stanford, Berkeley)

## Aim

- bridge the gap between theory and **practice**
- define the **state-of-the-art**


from YAHOO! LABS

### Music Recommendation Systems

- host: Yahoo!
- **11 years** of Yahoo! music data
- **2 tracks** of competition
- official dates: **March 15 to June 30**
- 1878 teams submitted to track 1;
  1854 teams submitted to track 2

# NTU Team for KDD Cup 2011

- 3 faculties:
  **Profs. Chih-Jen Lin, Hsuan-Tien Lin and Shou-De Lin**
- 1 course (similar to what was done in 2010):
  **Data Mining and Machine Learning: Theory and Practice**
- 3 TAs and 19 students:
  most were **inexperienced in music recommendation in the beginning**
- official classes: April to June;
  **actual classes: December to June**

our motto: study state-of-the-art approaches
and then **creatively improve them**

# The Track 1 Problem (1/2)

## Given Data

$263M$ examples (user $u$, item $i$, rating $r_{ui}$, date $t_{ui}$, time $\tau_{ui}$)

| user | item | rating | date | time |
|---|---|---|---|---|
| 1 | 21 | 10 | 102 | 23:52 |
| 1 | 213 | 90 | 1032 | 21:01 |
| 4 | 45 | 95 | 768 | 09:15 |
| ... | | | | |

- $u$, $i$: abstract IDs
- $r_{ui}$: integer between 0 and 100, **mostly multiples of** 10

## Additional Information: Item Hierarchy

- track (46.85%)
- album (19.01%)
- artist (28.84%)
- genre (5.30%)

# The Track 1 Problem (2/2)

## Data Partitioned by Organizers

- training: 253*M*; validation: 4*M*;
  test (w/o rating): 6*M*
- per user, **training < validation < test in time**
  - $\geq$ 20 examples total
  - 4 examples in validation; 6 in test
- **fixed random half of test: leaderboard**;
  **another half: award decision**

## Goal

predictions $\hat{r}_{ui} \approx r_{ui}$ on the test set, measured by

$$RMSE = \sqrt{\text{average}(\hat{r}_{ui} - r_{ui})^2}$$

note: one submission allowed **every eight hours**

# Three Properties of Track 1 Data

$\mathbf{R} =$

| | $\text{track}_1$ | $\text{track}_2$ | $\text{album}_3$ | $\text{author}_4$ | $\cdots$ | $\text{genre}_I$ |
|---|---|---|---|---|---|---|
| $\text{user}_1$ | 100 | 80 | 70 | ? | $\cdots$ | – |
| $\text{user}_2$ | – | 0 | ? | 80 | $\cdots$ | – |
| $\cdots$ | $\cdots$ | $\cdots$ | $\cdots$ | $\cdots$ | $\cdots$ | $\cdots$ |
| $\text{user}_U$ | ? | – | 20 | – | $\cdots$ | 0 |

*similar to Netflix data, but with the following differences......*

- scale: larger data

  training: study mature models that are computationally feasible

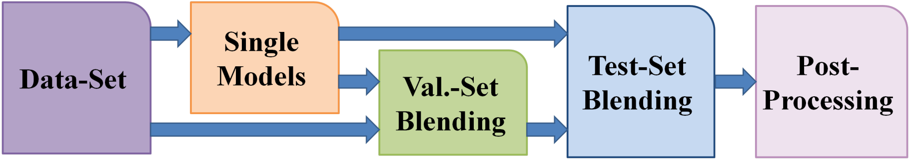
- taxonomy: relation graph of tracks, albums, authors and genres

  include as features for combining models nonlinearly

- time: detailed; training earlier than validation earlier than test

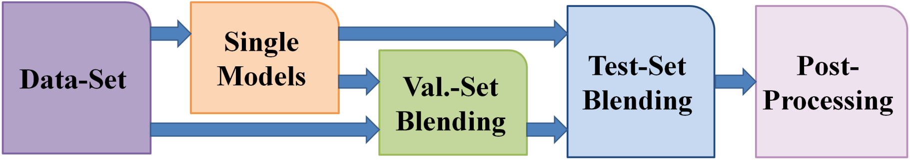
  include as features for combining models nonlinearly;
  respect time-closeness during training

# Framework of Our Solution




- single models—computationally feasible models that are diverse:
  - individual models: matrix factorization (& pPCA), pLSA
  - residual models: R-Boltz. machine, *k*-NN
  - derivative model: regression with statistical & model-based features
- validation-set blending:
  combine models nonlinearly while respecting time-closeness
- test-set blending:
  combine models linearly while fitting the leaderboard feedback
- post processing:
  polish predictions using findings during data analysis

# RMSE Performance at Each Stage of Framework




- single models: 22.7915
  - individual models: best RMSE 22.9022 (MF)
  - residual models: best RMSE 22.7915 ($k$-NN + MF)
  - derivative model: best RMSE 24.1251 (but helps in later stages)
- validation-set blending: 21.3598 [improvement 1.4317]
- test-set blending: (estimated) 21.0253 [improvement 0.3345]
- post processing: 21.0147 [improvement 0.0106]

both blending stages: key to the system

# Single Model: Matrix Factorization (1/2)

## Basic Idea

$$\mathbf{R} \approx \hat{\mathbf{R}} = \mathbf{P}^T\mathbf{Q} \text{ on the known examples}$$

- $\mathbf{P}^T$: $U$ (number of user) by $F$ (number of factor) user-factor matrix
- $\mathbf{Q}$: $F$ (number of factor) by $I$ (number of item) item-factor matrix
- one of the **most commonly-used** single models

## Training

- learn **P** and **Q** from data

$$\min_{\mathbf{P},\mathbf{Q}} \sum_{(u,i)\in\text{data}} \underbrace{(\hat{r}_{ui} - r_{ui})^2}_{E_{ui}(\cdot)} \text{ s.t. } \hat{r}_{ui} = \mathbf{p}_u^T\mathbf{q}_i$$

- large-scale optimization tool: stochastic gradient descent (SGD)
  1. randomly pick one example $(u, i)$
  2. $\mathbf{P} \leftarrow \mathbf{P} - \eta \cdot \nabla E_{ui}(\mathbf{P})$ (similar for $\mathbf{Q}$)

# Single Model: Matrix Factorization (2/2)

## Matrix Factorization Variants

$$\min_{\mathbf{P},\mathbf{Q},\cdots} \quad \text{regularization} + \sum_{(u,i)\in\text{data}} (\hat{r}_{ui} - r_{ui})^2$$

$$\text{s.t.} \quad \hat{r}_{ui} = \mathbf{p}_u^T \mathbf{q}_i + \bar{r} + \mu_u^{\text{user}} + \mu_i^{\text{item}} + \sigma_i \cdot \frac{\delta}{\delta + (t_{ui} - t_i^{\text{begin}})}$$

- extended terms (overall bias, user bias, item bias, time factor, etc.): enhance the power of model
- regularization: control the complexity of model
- parameter selection: tried **Automatic Parameter Tuning** tool

included many variants in the final solution for diversity

# Selected Ideas that Worked (1/5): Time Emphasis in Stochastic Gradient Descent

**Background**

SGD for minimizing sum of per-example $E_{ui}(\mathbf{P})$:

- randomly pick one example $(u, i)$
- $\mathbf{P} \leftarrow \mathbf{P} - \eta \cdot \nabla E_{ui}(\mathbf{P})$

**Idea**

- last $M$ steps of SGD: effectively considering only the last $M$ examples picked—**final P *as if* biased towards those**
- need: **P** respects time-closeness to the test examples
- heuristic: deterministically pick the "newer" examples as last

**consistent** $\approx$ 0.05 **RMSE improvement** for MF

# Single Model: Probabilistic Principle Component Analysis

## Basic Idea

$$P(r|u,i) = \mathcal{N}(\mathbf{p}_u^T \mathbf{r}_i + \bar{r}_u, \sigma^2)$$

- $\bar{r}_u$: user rating average
- can be viewed as probabilistic MF
- prediction $\hat{r}_{ui}$: expected rating over $P(r|u,i)$

## Training

- Expectation Maximization (EM) over maximum likelihood formulation

very similar to MF in the final solution

# Single Model: Probabilistic Latent Semantic Analysis

## Basic Idea

$$P(r|u,i) = \sum_{\kappa=1}^{k} P(r|i, z=\kappa) P(z=\kappa|u).$$

- $z$: the hidden variable representing user type
- can be viewed as **another way of factorization**
- prediction $\hat{r}_{ui}$: expected rating over $P(r|u,i)$

## Training

- basic: EM over maximum likelihood formulation
- improvement: **tempered EM**—EM + annealing (0.468 RMSE improvement)

not strong individually,
but quite different from MF solutions

# Residual Model: Restricted Boltzmann Machine

## Basic Idea

| discrete hidden factors | |
|---|---|
| $\uparrow$ | $\uparrow\downarrow$ |
| per-user incomplete discrete ratings | predicted continuous ratings |

- a recursive NNet
- popularly used in Netflix competition

## Training

- find the fixed point of the NNet weights
  by **contrastive divergence**

not better than MF individually,
but can be used to process residuals (see below)

# Selected Ideas that Worked (2/5): Gaussian RBM as Residual Model

## Background

- RBM: a recursive NNet; can be used as an individual model by

| discrete hidden factors | |
|---|---|
| $\uparrow$ | $\uparrow \downarrow$ |
| per-user incomplete discrete ratings | predicted continuous ratings |

- as individual: RMSE 24.7433, worse than MF (22.9974)

## Idea

- MF (a first-order model) efficiently gets better performance, but can RBM digest something different?
- need: RBM that learns from the residuals of MF $r_{ui} - \hat{r}_{ui}^{\text{MF}}$ (continuous values)

# Selected Ideas that Worked (2/5): Gaussian RBM as Residual Model

## Background

| discrete hidden factors | |
|---|---|
| $\uparrow$ | $\uparrow\downarrow$ |
| per-user incomplete discrete ratings | predicted continuous ratings |

## Idea

- need: RBM that learns from the residuals of MF
- choice: Gaussian RBM (gRBM)

| discrete hidden factors | |
|---|---|
| $\uparrow$ | $\uparrow\downarrow$ |
| per-user incomplete continuous residuals | predicted continuous residuals |

MF+gRBM: 22.8008;
better than individual MF (22.9974) or RBM (24.7433)

# Residual Model: $k$-Nearest Neighbor

## Basic Idea

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

- $G_k(u, i)$: item-neighbors of item $i$ (for user $u$)
- $w_{ij}$: correlation between items $i$ and $j$

## Training

- efficiently compute suitable neighbor and correlation functions

like RBM, not better than MF individually,
but can be used to process residuals

# Derivative Model: Regression

## Basic Idea

$$\hat{r}_{ui} = g(\mathbf{x}_{ui})$$

- $\mathbf{x}_{ui} \in \mathbb{R}^d$: some features related to user $u$ and item $i$
  - **statistical**: number of ratings from $u$, number of genre of item $i$, etc.
  - **model**: $\mathbf{p}_u$ in MF, $w_{ij}$ in $k$-NN, etc.
- $g$: a regressor from $\mathbb{R}^d$ to $\mathbb{R}$
  - linear regression
  - NNet
  - gradient boosting regression tree

can be flexibly used to include “side information” like hierarchy and time

# Glance of Single Model RMSE

| model | # used | best | average | worst | contribution |
|---|---|---|---|---|---|
| MF | 81 | 22.90 | 23.92 | 26.94 | **0.3645** |
| pPCA | 2 | 24.46 | 24.61 | 24.75 | 0.0014 |
| pLSA | 7 | 24.83 | 25.53 | 26.09 | 0.0042 |
| R-Boltz. machine | 8 | 22.80 | 24.75 | 26.08 | 0.0314 |
| *k*-NN | 18 | 22.79 | 25.06 | 42.94 | 0.0298 |
| regression | 10 | 24.13 | 28.01 | 35.14 | 0.0261 |

- contribution (**before val.-set blending**):
  estimated RMSE diff. via leave-the-model-out in test-set blending
- MF: most important (absorbing pPCA)
- residual models: both quite important
- derivative model: individually weak but adds diversity

val.-set blending:
95 models, best 21.36, average 23.53, worst 31.70

# Selected Ideas that Worked (3/5) in Val.-Set Blending: Multi-Feature and Multi-Stage Binned Lin. Reg.

## Background

- Binned Linear Regression: a conditional aggregation model
- different model strength on different "types" of examples
- different blending weights for different types (**bins**) to utilize strength

| bins | # rating $\leq \theta_1$ | $\theta_1 <$ # rating $\leq \theta_2$ | others |
|---|---|---|---|
| weight of MF-1 | 0.4 | 0.7 | 1.0 |
| weight of RBM-1 | 0.5 | 0.1 | 0.0 |
| weight of RBM-2 | 0.1 | 0.2 | 0.0 |

- a simplified regression tree with one level (on one feature)

# Selected Ideas that Worked (3/5) in Val.-Set Blending: Multi-Feature and Multi-Stage Binned Lin. Reg.

## Background

- Binned Linear Regression
  —different blending weights for different (types) bins of examples

## Idea: **multi-feature** BLR

- rationale: “type” more sophisticated than 1-feature bin
- a special multi-level decision tree
- prevent overfitting by limiting height and bin size
- heuristic algorithm instead of traditional decision tree:
  due to simplicity by extending from one-feature BLR

| multi-feature | 1-feature | 4-feature | 6-feature |
|---|---|---|---|
| RMSE | 22.0829 | 21.8605 | 21.8128 |

# Selected Ideas that Worked (3/5) in Val.-Set Blending: Multi-Feature and Multi-Stage Binned Lin. Reg.

## Background

- Binned Linear Regression
  —different blending weights for different (types) bins of examples

## Idea: **multi-stage** BLR

- rationale: more diverse but good models before test-set blending

| bins | 1 | 2 | 3 |
|---|---|---|---|
| weight of MF-1 | ... | ... | ... |
| weight of RBM-1 | ... | ... | ... |
| weight of RBM-2 | ... | ... | ... |
| weight of BLR-1 | ... | ... | ... |
| weight of BLR-2 | ... | ... | ... |

| multi-stage | 1-stage | 2-stage | 3-stage |
|---|---|---|---|
| RMSE | 21.7140 | 21.4591 | 21.4287 |

# Selected Ideas that Worked (4/5) in Test-Set Blending: Offline Test Performance Predictor

## Background

- given: columns $\mathbf{z}_m$ = test-set prediction of model $m$
- test-set linear regression:

$$\mathbf{w}(\mathbf{z}_1, \mathbf{z}_2, \cdots, \mathbf{z}_M, \lambda) = (\mathbf{Z}^T\mathbf{Z} + \lambda\mathbf{I})^{-1}\mathbf{Z}^T\mathbf{r}$$

- true ratings $\mathbf{r}$ unknown but $\mathbf{z}^T\mathbf{r}$ can be estimated by

$$\begin{aligned} 2\mathbf{z}^T\mathbf{r} &= \mathbf{z}^T\mathbf{z} + \mathbf{r}^T\mathbf{r} - (\mathbf{z} - \mathbf{r})^T(\mathbf{z} - \mathbf{r}) \\ &\approx \mathbf{z}^T\mathbf{z} + N \cdot \mathrm{RMSE}(\mathbf{0})^2 - N \cdot \mathrm{RMSE}(\mathbf{z})^2 \end{aligned}$$

- common technique for RMSE ever since Netflix competition

# Selected Ideas that Worked (4/5) in Test-Set Blending: Offline Test Performance Predictor

## Background

$$\begin{aligned} 2\mathbf{z}^T\mathbf{r} &= \mathbf{z}^T\mathbf{z} + \mathbf{r}^T\mathbf{r} - (\mathbf{z} - \mathbf{r})^T(\mathbf{z} - \mathbf{r}) \\ &\approx \mathbf{z}^T\mathbf{z} + N \cdot \mathrm{RMSE}(\mathbf{0})^2 - N \cdot \mathrm{RMSE}(\mathbf{z})^2 \end{aligned}$$

## Idea

- want: decide which $\mathbf{z}_m$'s and $\lambda$ to use
- restriction: one submission every eight hours
- solution: estimate RMSE of $\mathbf{w}$ without submitting more than $\mathbf{z}_m$

$$N \cdot \mathrm{RMSE}(\mathbf{w})^2 = (\mathbf{Z}\mathbf{w} - \mathbf{r})^T(\mathbf{Z}\mathbf{w} - \mathbf{r}) = \mathbf{w}^T\mathbf{Z}^T\mathbf{Z}\mathbf{w} - 2\mathbf{w}^T\mathbf{Z}^T\mathbf{r} + \mathbf{r}^T\mathbf{r}$$

compute the contribution of models;
choose 221 from $\approx$ 300 models & decide $\lambda = 10^{-6}$ offline

# Selected Ideas that Worked (5/5) in Post-Processing: Clipping for Old Four-Star Days

## Background

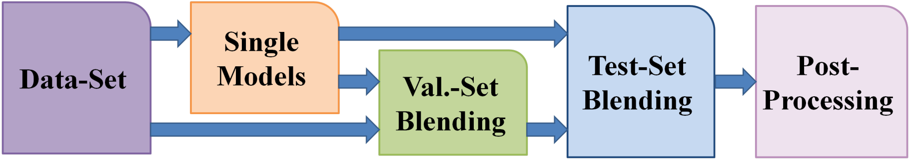
- some very different rating systems observed during data analysis:
  - four-star rating? $\{0, 30, 50, 70, 90\}$
  - five-star rating? $\{0, 20, 40, 60, 80, 100\}$
  - 100-point scale
- suspect changes in the user interface of Yahoo! Music

## Idea

- existing: in five-star or 100-point scale, clip prediction to $[0, 100]$
- new: for four-star, clip prediction to $[0, 90]$
- what dates? $[3365, 5982]$ (7 years) or $[4281, 6170]$ (5 years)

$\approx$ 0.02 RMSE improvement on most models

# Revisit: RMSE Performance at Each Stage of Framework




- single models: 22.7915
  - individual models: best RMSE 22.9022 (MF)
  - residual models: best RMSE 22.7915 ($k$-NN + MF)
  - derivative model: best RMSE 24.1251 (but helps in later stages)
- validation-set blending: 21.3598 [improvement 1.4317]
- test-set blending: (estimated) 21.0253 [improvement 0.3345]
- post processing: 21.0147 [improvement 0.0106]

both blending stages: key to the system

# Selected Ideas that Did Not Work: Deal with Zero-Variance Users

## Background

- zero-variance users (7% of all users)
  —if a user gives 60, 60, 60, $\cdots$ in all training ratings, how'd she rate the next item?
- Occam's razor prediction: 60
  —only true for 80% of users, 20% changed their mind!

## Idea

- conditionally (the 80%) post-process the predictions
- difficult to distinguish and thus failed

# Track 1 Mini-Summary

- individual models
  - single: **MF** (& pPCA), pLSA
  - residual: RBM, *k*-NN
  - derivative: regression

—concept of diversity important

- blending
  - validation: deeply and non-linear to improve model power
  - test: broadly and linear to use leaderboard feedback properly (**with good estimation**)

Next: Track 2 by Prof. Shou-De Lin