

# Sistemas Recomendadores

IIC-3633

Ensamblés Parte 2

# Esta clase

1. Recomendación basada en ensambles basada en estrategias de mezcla.

# Combining Predictions for Accurate Recommender Systems

Michael Jahrer  
commendo  
research&consulting, Graz  
University of Technology  
8580 Köflach  
Austria  
michael.jahrer@  
commendo.at

Andreas Töschler  
commendo  
research&consulting, Graz  
University of Technology  
8580 Köflach  
Austria  
andreas.toeschler@  
commendo.at

Robert Legenstein  
Institute for Theoretical  
Computer Science, Graz  
University of Technology  
8010 Graz  
Austria  
robert.legenstein@igi.tugraz.at

# Introducción

- Si bien es posible hacer competir diferentes métodos, el artículo de Jahrer et al. indica que hay beneficios en mezclar diferentes recomendadores.
- Los autores lo usaron de forma efectiva en el Netflix Prize.

# Modelos

El artículo describe varios modelos para la tarea de predicción de rating y los combina utilizando distintas estrategias:


- **KNNuser**
- **KNNitem**
- **SVD**
- **SVD<sub>e</sub>**: SVD extendido que incluye timestamp como feature adicional.
- **Asymmetric Factor Model (AFM)**, SVD se enfoca solo en item features.
- **Restricted Boltzmann Machines (RBM)**
- **General Effects (GE)**: estadísticas generales, ej. rating promedio.

# Resultados

Obtienen que la mezcla de los métodos (Blend) obtiene error  $< 0.87$ .


No hay mejoras significativas pero les permitió ganar el concurso.


algorithm	RMSE (approx.)
KNNitem	0.92
KNNuser	0.93
SVD	0.90
AFM	0.92
SVD <sub>e</sub>	0.88
RBM	0.90
GE	0.95
Blend	$<0.87$

 Featured Code Competition

## Optiver - Trading at the Close

Predict US stocks closing movements

 Optiver · 520 teams · 3 months to go (3 months to go until merger deadline)



### \$100,000

Prize Money

[Overview](#) [Data](#) [Code](#) [Models](#) [Discussion](#) [Leaderboard](#) [Rules](#)

[Join Competition](#) ...

## Overview

In this competition, you are challenged to develop a model capable of predicting the closing price movements for hundreds of Nasdaq listed stocks using data from the order book and the closing auction of the stock. Information from the auction can be used to adjust prices, assess supply and demand dynamics, and identify trading opportunities.

### Start

8 days ago



### Close

3 months to go

Merger & Entry

### Competition Host

Optiver



### Prizes & Awards

\$100,000

Awards Points & Medals

### Participation

588 Competitors

520 Teams

2,426 Entries

# Todas las versiones probadas (19 modelos en total)

nr	name	RMSE	description
1	AFM-1	0.9362	AFM, 200 features, $\eta = 1e-3$ , $\lambda = 1e-3$ , learnrate $\eta$ 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 $\eta$ and regularization constants $\lambda$ 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.



# Alternativas de mezcla

- El artículo describe varias alternativas de “blending”:
  - Linear regression
  - Binned linear regression
  - Neural Network
  - Bagged Gradient Boosted Decision Tree
  - Kernel ridge regression blending
  - K-Nearest Neighbors

# Linear Regression

- Predice el rating basándose en la suma ponderada de predicciones de ratings de cada algoritmo.
- Busca la combinación óptima de diferentes sistemas.

the prediction is  $\Omega(\mathbf{x}) = \mathbf{x}^T \mathbf{w}$ . Weights  $\mathbf{w}$  are calculated with ridge regression,  $\mathbf{w} = (\mathbf{X}^T \mathbf{X} + \lambda \mathbf{I})^{-1} \mathbf{X}^T \mathbf{y}$ , where  $\mathbf{I}$  denotes the identity matrix. Cross-validation is used in order to select a proper ridge regression constant  $\lambda$ .

donde:

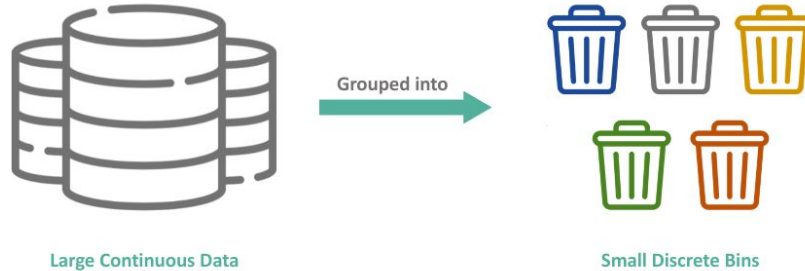
$\mathbf{X}$  = predicciones de ratings de cada modelo.

$\mathbf{w}$  = pesos calculados usando una ridge regression

- Las predicciones de rating de cada sistema son características de entrada.
- Aprende pesos óptimos para combinar estas predicciones en un rating final único.

# Binned linear regression

- Divide los datos en segmentos o "bins" y ajusta un modelo diferente a cada uno de estos segmentos.



# Criterios para dividir en bins

- Support : The support of a data point  $(u, i)$  is the number of votes by user  $u$ . The blender can now base the weighting of predictors dependent on how many rating the user has given. RBMs are prone to receive high weight when the user has only a few votes in the data. SVDs are highly weighted when much information from a user is available.
- Time : The time of a data point  $(u, i)$  is the day on which the rating  $r_{ui}$  was performed. Predictions are mixed together with time dependency. When using this binning criteria, the blender can easily model time-dependent blending.
- Frequency : The frequency of a data point  $(u, i)$  is the number of ratings from a user at day  $t$ . This criteria enables the blender to be selective, based on the user's rating day frequency. The blender has the ability to give predictions other weights when a user votes many times on a particular day.

Cantidad de interacciones

Día en que se dio el rating

Frecuencia de ratings en un día determinado

## Ejemplo de división en bins (support)

Here,  $b$  is in general chosen based on information about user  $u$  and item  $i$ . For example, if we divide the Netflix probe-set into 5 support-bins, the following formula gives the bin  $b$ :

$$b = \begin{cases} 1, & |N(u)| < 34 \\ 2, & 35 \leq |N(u)| < 70 \\ 3, & 71 \leq |N(u)| < 146 \\ 4, & 147 \leq |N(u)| < 321 \\ 5, & \text{else,} \end{cases} \quad (3)$$

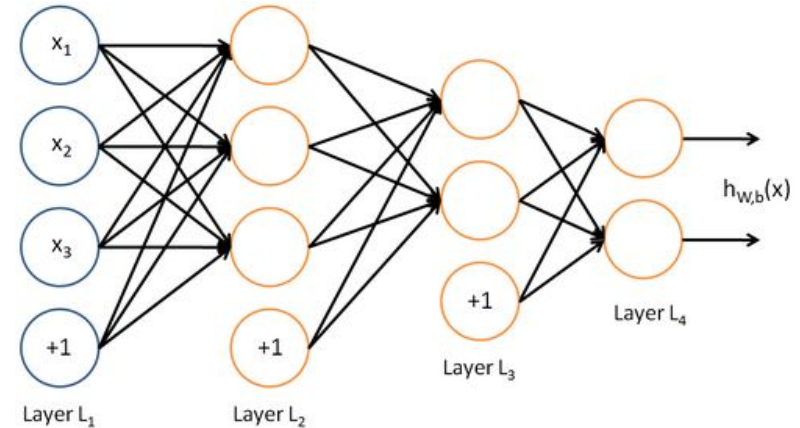
donde  $|N(u)|$  es el número de ratings dados por el usuario ( $u$ )

- Las predicciones de rating se agrupan en bins según sus características.
- Se entrena un modelo de regresión lineal diferente para cada bin.
- La recomendación final se genera a partir de los modelos correspondientes a los bins, ponderando sus predicciones.



# Neural Network

- Entrena una red neuronal que aprende un peso para cada modelo dada su predicción.
- Captura relaciones no lineales entre los modelos y sus predicciones.



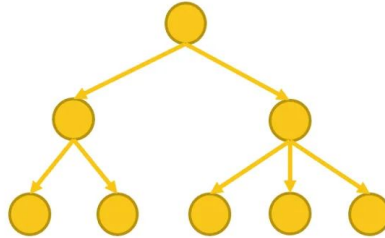
- Las predicciones de rating de diferentes sistemas se utilizan como entrada.
- Aprende una función no lineal para combinar estas predicciones.
- La salida de la red neuronal es el rating final, una combinación no lineal de las predicciones iniciales.

# Bagged Gradient Boosted Decision Tree

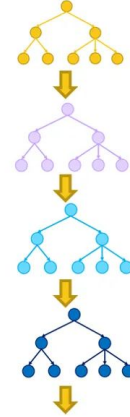
Usa bagging para reducir el sobreajuste.

Construye árboles secuencialmente para corregir errores aprendiendo de las diferencias del árbol anterior.

Single Decision Tree

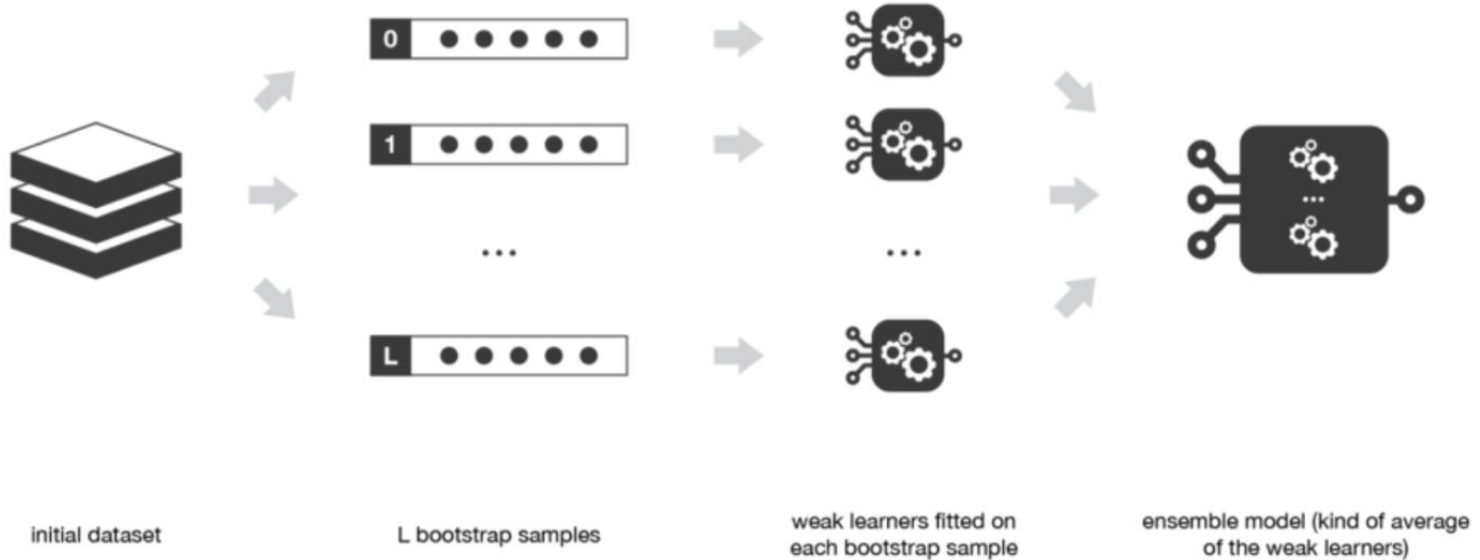


Gradient Boosted Trees

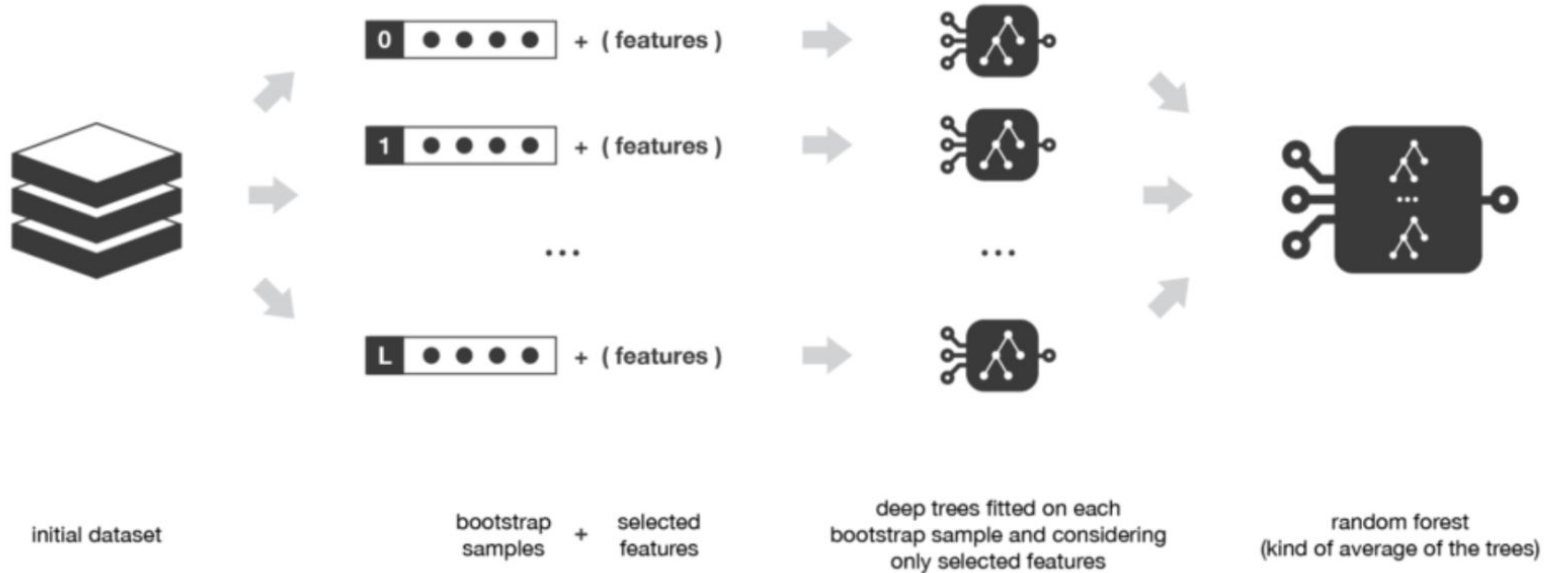


- Crea múltiples subconjuntos de datos de entrenamiento con predicciones de rating.
- Entrena un árbol en cada uno y promedia sus predicciones para obtener el rating final.

# Bagging



# Bagging – Random Forests

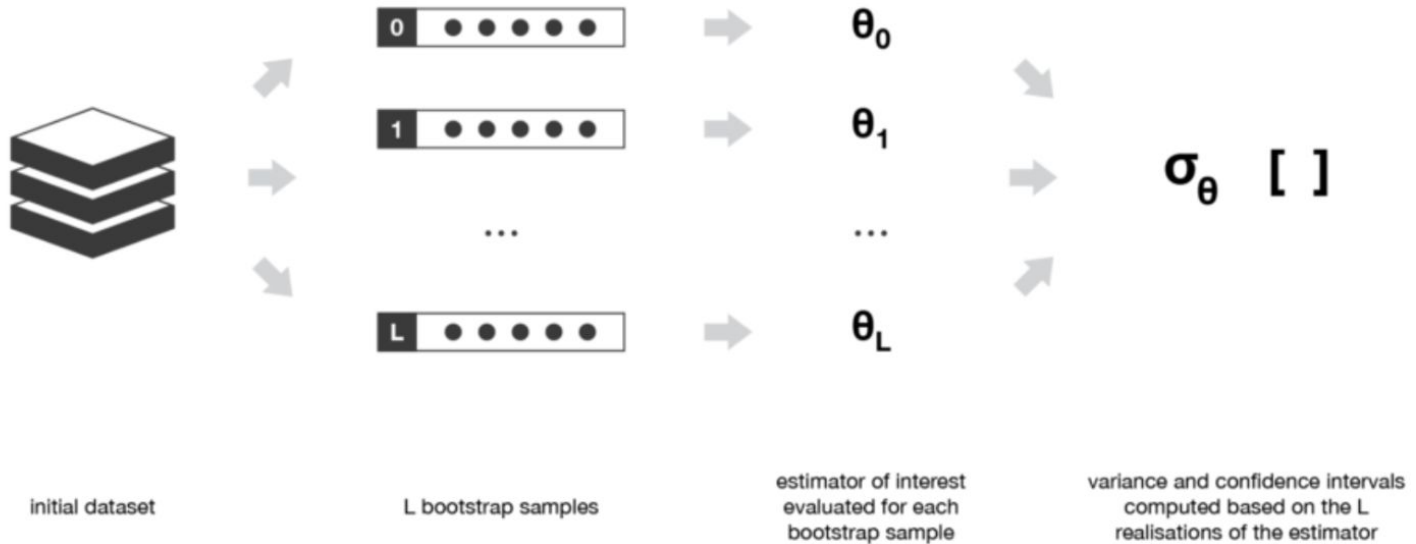


Agrega "selected features"

# Boosting

- **Secuencial:** A diferencia del bagging, en boosting los modelos se construyen secuencialmente.
- **Corrige Errores:** Cada modelo nuevo intenta corregir los errores cometidos por el conjunto de modelos anteriores.
- **Ponderación:** Los modelos que tienen buen rendimiento tienen más peso en la votación final.

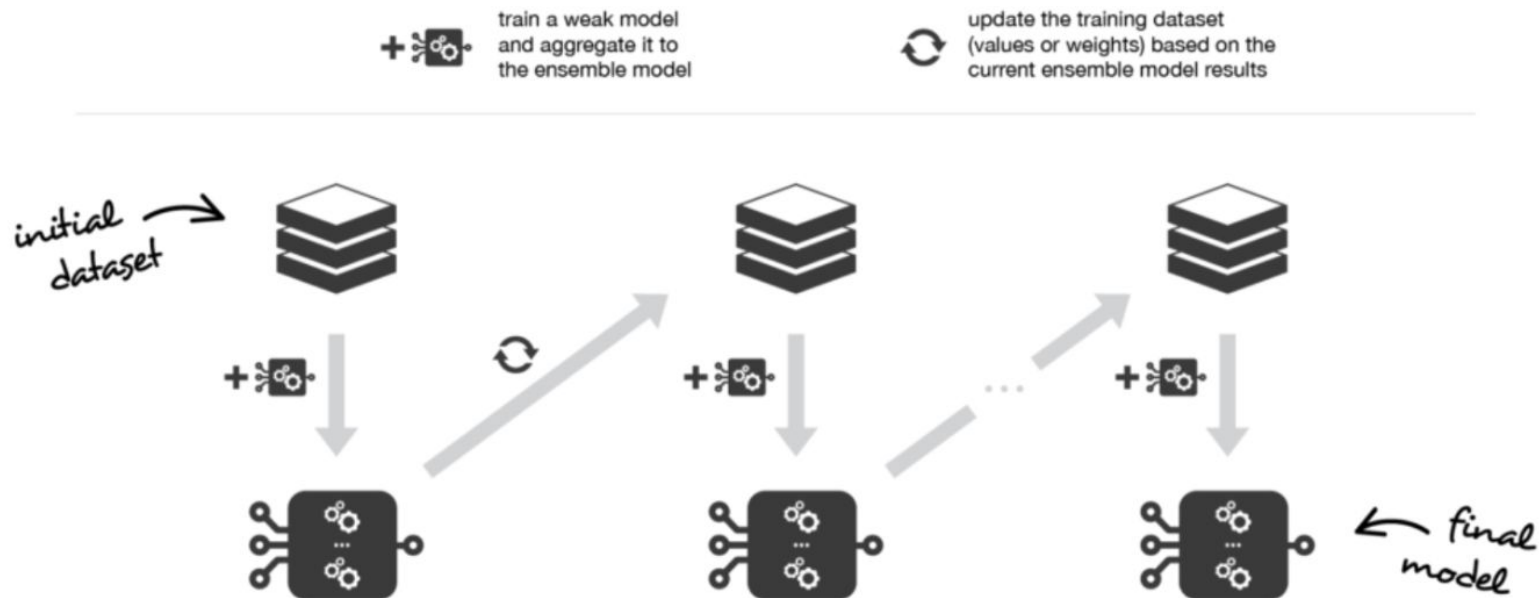
# Boosting










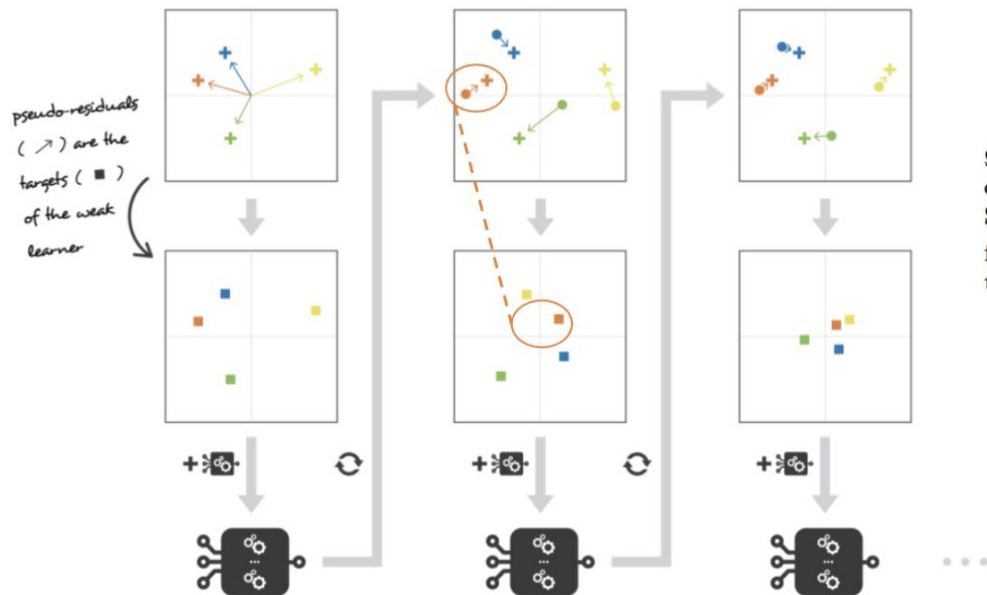
# Boosting como ML

$\Omega_1$  in the chain is trained on the unmodified targets  $\mathbf{y}_1 = \mathbf{y}$  of the dataset. The second model trains on  $\mathbf{y}_2 = \mathbf{y} - \eta \mathbf{y}_1$ . So targets in each cascade are  $\mathbf{y}_i = \mathbf{y} - \sum_{j=1}^{i-1} \eta \mathbf{y}_j$ . The final prediction model is  $\Omega(\mathbf{x}) = \sum_{i=1}^M \eta \Omega_i(\mathbf{x})$ , where M is the length of the boosting chain.



# Gradient Boosted Trees

-  train a weak model and aggregate it to the ensemble model
-  update the pseudo-residuals considering predictions of the current ensemble model
-  dataset values
-  predictions of the current ensemble model
-  pseudo-residuals (targets of the weak learner)



$\Omega_1$  in the chain is trained on the unmodified targets  $\mathbf{y}_1 = \mathbf{y}$  of the dataset. The second model trains on  $\mathbf{y}_2 = \mathbf{y} - \eta \mathbf{y}_1$ . So targets in each cascade are  $\mathbf{y}_i = \mathbf{y} - \sum_{j=1}^{i-1} \eta \mathbf{y}_j$ . The final prediction model is  $\Omega(\mathbf{x}) = \sum_{i=1}^M \eta \Omega_i(\mathbf{x})$ , where  $M$  is the length of the boosting chain.

# En resumen

**Bagging** construye modelos independientemente y los combina.

**Boosting** construye modelos secuencialmente, donde cada modelo intenta corregir los errores del anterior.

Ambas técnicas se utilizan para mejorar la precisión y reducir el sobreajuste, pero operan de manera diferente.

El Bagged Gradient Boosted Decision Tree hace uso de ambas técnicas en conjunto.

# Kernel ridge regression blending

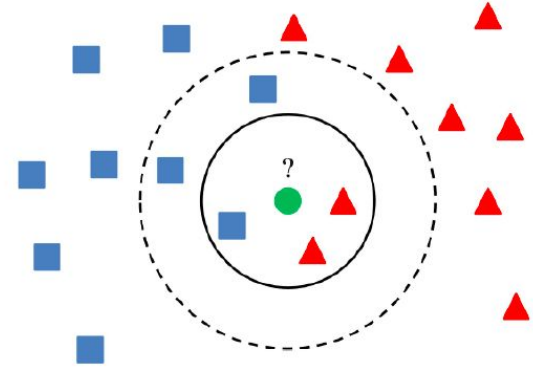
- Aplica una combinación lineal en un espacio de alta dimensión.
- Facilita la detección de relaciones lineales.

The learning algorithm is described in [20]. The Gauss kernel  $k(\mathbf{x}, \mathbf{y}) = \exp(-(\|\mathbf{x} - \mathbf{y}\|^2)/\sigma^2)$  is used in our experiments. The width  $\sigma$  of the kernel and the regularization parameter  $\lambda$  are tuned with cross validation.

- Mapea las predicciones de rating a un espacio de mayor dimensión mediante el truco del kernel.
- Aplica regresión lineal en este espacio transformado para obtener pesos óptimos.
- El rating final es una combinación ponderada de las predicciones iniciales, considerando las relaciones complejas en el espacio transformado.

# K-Nearest Neighbors

- Se basa en la similitud entre ejemplos.
- Utiliza la similitud entre listas de recomendación.
- Para cada ítem, crea un vector de características donde cada característica es la puntuación dada por cada recomendador.
- Entrena un KNN utilizando estos vectores .



Cuando quieras hacer una predicción para un ítem, usa el vector de características de ese ítem para buscar los  $k$  vecinos más cercanos utilizando el modelo KNN entrenado.

Una vez que tienes los  $k$  vecinos más cercanos, puedes hacer la predicción final agregando las etiquetas de estos vecinos (ej. promedio, ponderar por distancia, etc...)



# Resultados

# Resultados - LR

-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

**Table 3: Blending weights of an optimal linear combination. This leads to 0.875258 RMSE on the pTest set. The RMSE on the cross validation set is 0.87552. The number in the brackets is the probe RMSE per model.**

## Resultados - Binned LR

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

**Table 4: RMSE values obtained with binned linear regression on the pTest set. The small values in the brackets below are RMSEs from the cross validation.**

# Resultados - NN

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

19 input  
30 neuronas ocultas  
1 output

**Table 5:** Results from different neural network blends. We use in all networks the same learning rate  $\eta = 5e-4$ ,  $\eta^{(-)} = 5e-7$ . Output transformation constants are  $\alpha = 3.6$ ,  $\beta = 3.0$ .

# Resultados - BGBDT

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

We analyzed Bagged Gradient Boosted Decision Trees with varying bagging size  $N_{bag}$ , varying subspace size  $S$ , varying number of leaves  $K$ , and varying learning rate  $\eta$ .

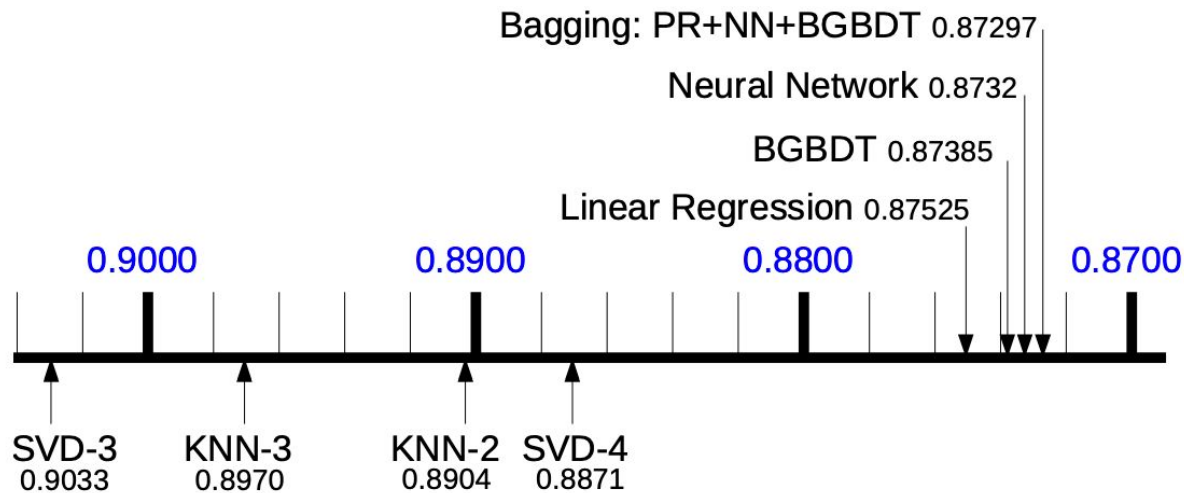
**Table 6: BGBDT blending results.** The first column denotes the fixed parameters. In the next columns the first line is the tested parameters, second line is the validation RMSE, third line is the pTest RMSE and fourth line is the training time. We vary the learn rate  $\eta$ , the subspace size  $K$  and the bagging size  $N_{bag}$ . For all results we use optimal splits in training a single tree.

# Resultados – Bag with NN, LR, GBDT

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]

**Table 7: Bagging and linear combination of many blending models applied to collaborative filtering for the Netflix Prize dataset. The first column indicates the model type. The second column reports the individual out-of-bag RMSE estimate, the number in brackets below is the blend RMSE. The third column shows the weight of the model. The fourth column shows metaparameters and the training time of each model. Accurate blending methods receive higher weights.  $N_{bag} = 128$  in all models.**





**Figure 5: RMSE values on the Netflix Prize probe set. Shown are various blending methods (above the scale) and the best performing single algorithms (below the scale).**

## Conclusions

For practical applications we recommend to use a neural network in combination with bagging due to the fast prediction speed. A set of  $10^6$  samples can be predicted in a few seconds with this technique. We showed that a large ensemble of different collaborative filtering models leads to an accurate prediction system.



Gracias!