

NETFLIX

*Recommender Systems
Real world ML*

Lyle Ungar
University of Pennsylvania

The Netflix training data

◆ Training data

- 480,000 users
- 18,000 movies
- 100,000,000 ratings

\$1,000,000 prize money
for first team to beat the
baseline by 10%

◆ Data is sparse

- $100,000,000 / (18,000 * 480,000) = 0.01$
- but it is worse than that!

Validation and Test data sets

- ◆ **Validation (“Quiz”) set**

- 1.4 million ratings used to calculate leaderboard

- ◆ **Test set**

- 1.4 million ratings used to determine winners

What models to use?

- ◆ An ensemble of many models
- ◆ Main methods
 - K-nearest neighbors
 - Matrix reconstruction

K-NN

$$\hat{r}_{ui} = \sum_{j \in N(i;u)} r_{uj} / k$$

r_{ui} = rating by user u for movie i

$N(i;u)$ = the set of k (typically 20–50) movies for which user u has provided a rating that are most similar to movie i

How do you measure movie similarity?

How to improve?

?

Top

Soft K-NN

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

r_{ui} = rating by user u for movie i

s_{ij} = similarity between movies i and j

K-NN – subtract off a baseline

$$\hat{r}_{ui} = b_{ui} + \sum_{j \in N(i;u)} s_{ij} (r_{uj} - b_{uj}) / \sum_{j \in N(i;u)} s_{ij}$$

r_{ui} = rating by user u for movie i

s_{ij} = similarity between movies i and j

b_{ui} = baseline rating - e.g. mean rating of user u or movie i

This doesn't account for

- ◆ Similar movies are redundant
 - e.g. a series like Star Wars or Avengers
- ◆ Movies may be more or less similar
 - If less similar, then 'shrink' more to the baseline

K-NN with regression instead of similarity

$$\hat{r}_{ui} = b_{ui} + \sum_{j \in N(i; u)} w_{ij} (r_{uj} - b_{uj})$$

r_{ui} = rating by user u for movie i

w_{ij} = weight learned by regression

b_{ui} = baseline rating - e.g. mean rating of user u or movie i

K-NN with regression

$$\hat{r}_{ui} = b_{ui} + \sum_{j \in N(i; u)} w_{ij} (r_{uj} - b_{uj})$$

r_{ui} = rating by user u for movie i

w_{ij} = weight learned by regression

b_{ui} = baseline rating - e.g. mean rating of user u or movie i

Find w_{ij} by seeing what weights on similar movies j would have best estimated the rating r_{vi} on the target movie i by people v other than the user u .

$$\operatorname{argmin}_w [\sum_{v \neq u} (r_{vi} - \hat{r}_{vi})^2] = \operatorname{argmin}_w [\sum_{v \neq u} (r_{vi} - b_{vi} - \sum_{j \in N(i; v)} w_{ij} (r_{vj} - b_{vj}))^2]$$

This can be expensive

- ◆ Need to compare every user against every other user to find the most similar users
 - Based on movies in common
- ◆ How to speed up?

What is the answer?

Top

Matrix factorization

- ◆ Factor rating matrix \mathbf{R}
- ◆ $\hat{\mathbf{R}} = \mathbf{P}\mathbf{Q}^T$ or $\hat{r}_{ui} = \mathbf{p}_u \mathbf{q}_i^T$
 - \mathbf{P} is number of users * number of hidden factors
 - \mathbf{Q} is number of movies * number of hidden factors
 - Number of hidden factors, $k = 60$
- ◆ \mathbf{P} looks like principal component scores/coefficients
- ◆ \mathbf{Q} looks like loadings

Matrix factorization

$$\sum_{(u,i) \in K} [(r_{ui} - p_u q_i^T)^2 + \lambda (\|p_u\|_2^2 + \|q_i\|_2^2)]$$

reconstruction error ridge penalty

where the summation is over the set K of (u,i) pairs for which r_{ui} are known.

◆ Solve using alternating least squares

- first fix P and solve for Q using Ridge regression
- then fix Q and solve for P using Ridge regression
- repeat.

Matrix factorization

- ◆ Further *regularize* by forcing the elements of P and Q to be non-negative
 - Non-Negative Matrix Factorization (NNMF)
- ◆ And do locally weighted matrix factorization
 - $\sum_{(u,i) \in K} [s_{ij}(r_{ui} - p_u q_i^T)^2 + \lambda(|p_u|_2 + |q_i|_2)]$

What is out-of-sample?

What you should know

- ◆ **Everything we've done can be extended to only use the loss over the observations we have.**
 - Regression
 - Matrix factorization
 - Generalizes PCA to include Ridge penalty, NNMF
- ◆ **For highest accuracy, ensemble all the methods**
 - Subtract off (and add back in) baseline
- ◆ **Follow-up competition was cancelled because...**
- ◆ **Lots of other features can be used**

How is my speed?

Slow

Good

Fast