西安交通大学
XI'AN JIAOTONG UNIVERSITY

机器学习在信息安全中的应用

言浬
li.yan.88@xjtu.edu.cn

2025年11月

小测验

- 下列关于梯度下降法描述正确的是？（A,B,C）

A. 可以用于求解逻辑斯蒂回归

B. 是一种迭代求解的方法

C. 可以比较好的并行化

D. 可以高效地求解所有凸优化问题

小测验

- 商家想利用天气来预测销售额，已知他只考虑温度、湿度、降雨量三种天气特征。若商家使用线性回归模型预测，则模型的输入是（C）维的。

A. 1

B. 2

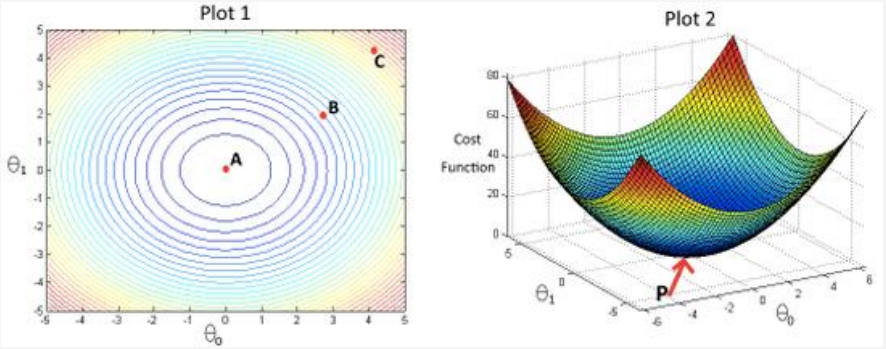
C. 3

D. 4

小测验

- 代价函数$J(\theta_0,\theta_1)$与θ_0，θ_1的关系如图2所示。“图1”中给出了相同代价函数的等高线图。根据图示，选择正确的选项

A, E



A. 从B点开始，学习率合适的梯度下降算法会最终帮助我们到达或者接近A点，即代价函数 $J(\theta_0,\theta_1)$ 在A点有最小值

B. 点P（图2的全局最小值）对应于图1的点C

C. 从B点开始，学习率合适的梯度下降算法会最终帮助我们到达或者接近C点，即代价函数$J(\theta_0,\theta_1)$在C点有最小值

D. 从B点开始，学习率合适的梯度下降算法会最终帮助我们到达或者接近A点，即代价函数$J(\theta_0,\theta_1)$在A点有最大值

E. 点P（图2的全局最小值）对应于图1的点A

小测验

- 假设对于某个线性回归问题（比如预测房价），我们有一些训练集，对于我们的训练集，我们能够找到一些 θ_0, θ_1，使得 $J(\theta_0, \theta_1)$=0。以下哪项陈述是正确的?

B

A. 为了实现这一点，我们必须有 θ_0=0，θ_1=0，这样才能使 $f_\theta(x_i)$=0

B. 对于满足 $J(\theta_0, \theta_1)$=0的 θ_0, θ_1 的值，其对于每个训练例子 (x_i, y_i)，都有 $f_\theta(x_i) = y_i$

C. 不可能存在 θ_0, θ_1，使得 $J(\theta_0, \theta_1)$=0

D. 即使对于我们还没有看到的新例子，我们也可以完美地预测y的值（例如，我们可以完美地预测我们尚未见过的新房的价格）

小测验

- 假设已经训练了一个logistic分类器，它在一个新示例x上输出一个预测 $f_\theta(x)$ =0.4。这意味着：

C，D

A. 我们对 P(y=0 | x) 的估计是0.4

B. 我们对 P(y=1 | x) 的估计是0.6

C. 我们对 P(y=0 | x) 的估计是0.6

D. 我们对 P(y=1 | x) 的估计是0.4

小测验

- 以下哪项陈述是正确的？选出所有正确项。 B, D

A. 在构建学习算法的第一个版本之前，花大量时间收集大量数据是一个好主意。

B. 在分布不均衡的数据集上（例如，当正例远多于负例时），精确率、召回率、F1分数都不是很好的性能度量，应该用AUC。

C. 训练完逻辑回归分类器后，必须使用0.5作为判断阈值。

D. 使用一个非常大的训练集使得模型不太可能过度拟合训练数据。

E. 如果模型欠拟合，获取更多数据可能会有帮助。

小测验

- 用η=0.3进行15次梯度下降迭代，每次迭代后计算$J(\theta)$，发现其下降缓慢，并且在15次迭代后仍在下降。基于此，以下哪个结论似乎最可信？ C

A. η =0.3是学习率的有效选择

B. 与其使用η当前值，不如尝试更小的η值（比如η =0.1）

C. 与其使用η当前值，不如尝试更大的η值（比如η =1.0）

小测验

- 假设有n=14个数据实例，有d=3个特征（不包括需要另外添加的恒为1的截距项），线性回归的矩阵形式为$\hat{\boldsymbol{y}} = \boldsymbol{X}\boldsymbol{\theta}$。对于给定n和d，$\boldsymbol{X}$，$\hat{\boldsymbol{y}}$，$\boldsymbol{\theta}$的维度分别是多少？

B

A. $\boldsymbol{X}$ 14×3，$\hat{\boldsymbol{y}}$ 14×1，$\boldsymbol{\theta}$ 3×3

B. $\boldsymbol{X}$ 14×4，$\hat{\boldsymbol{y}}$ 14×1，$\boldsymbol{\theta}$ 4×1

C. $\boldsymbol{X}$ 14×3，$\hat{\boldsymbol{y}}$ 14×1，$\boldsymbol{\theta}$ 3×1

D. $\boldsymbol{X}$ 14×4，$\hat{\boldsymbol{y}}$ 14×4，$\boldsymbol{\theta}$ 4×4

第三章：k近邻算法

3.1 k近邻算法

3.2 基于近邻算法的协同过滤

3.3 矩阵分解

西安交通大学

XI'AN JIAOTONG UNIVERSITY

3.1 k近邻算法

3.1 k近邻算法

k近邻算法（k-Nearest Neighbor Algorithm）

- 用于分类和回归的非参数方法
  - 对于每个输入实例x，在特征空间中找到k个最接近的训练实例$N_k(x)$
  - 对于回归问题：对x的预测基于k个实例的标签的平均值

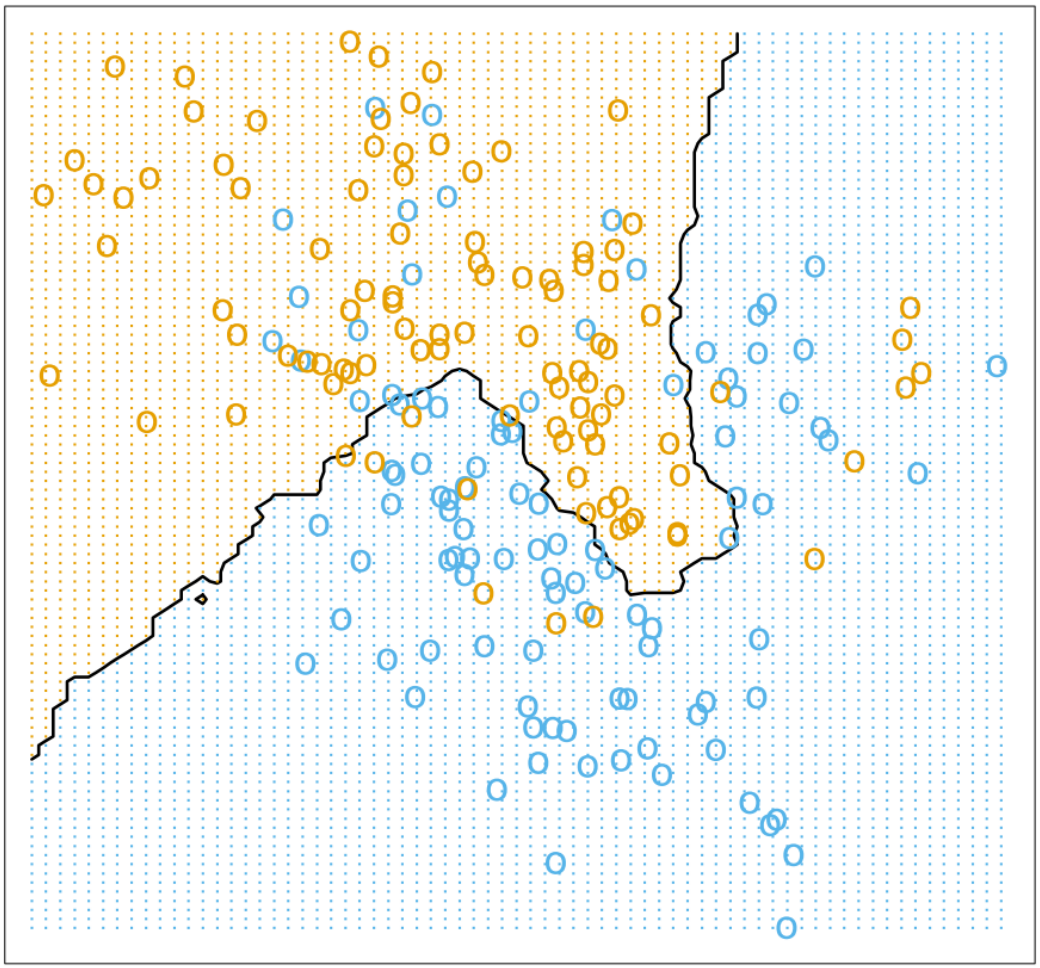
$$\hat{y}(x) = \frac{1}{k} \sum_{x_i \in N_k(x)} y_i$$

  - 对于分类问题：做邻居间的多数投票

$$\hat{y}(x) = \text{majority vote } \{y_i, x_i \in N_k(x)\}$$

3.1 k近邻算法

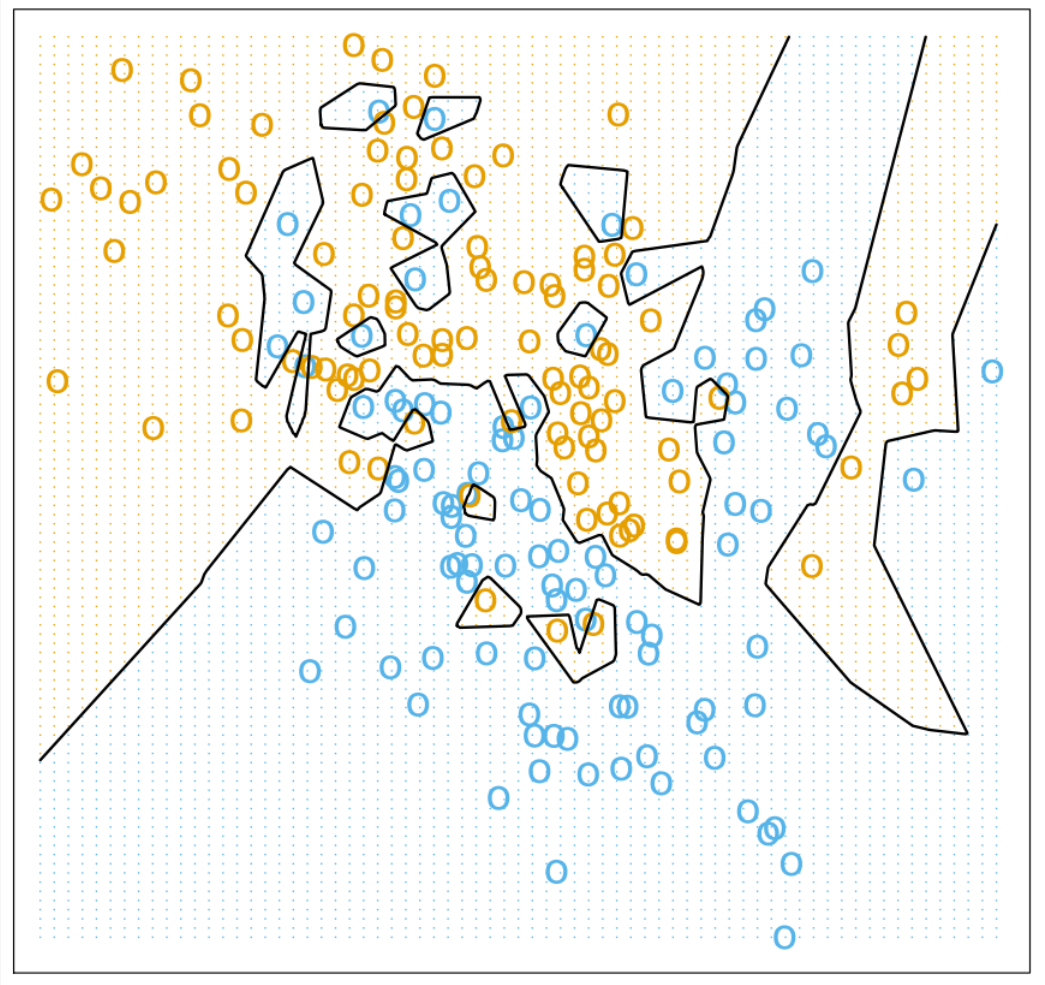
15-近邻



Jerome H. Friedman, Robert Tibshirani, and Trevor Hastie. “The Elements of Statistical Learning”. Springer 2009.

3.1 k近邻算法

1-近邻



Jerome H. Friedman, Robert Tibshirani, and Trevor Hastie. “The Elements of Statistical Learning”. Springer 2009.

3.1 k近邻算法

k近邻算法（k-Nearest Neighbor Algorithm）

- 广义版本
  - 定义输入实例x与其邻居x_i之间的相似度函数$s(x, x_i)$
  - 然后基于邻近相似性的加权标签平均值进行预测

$$\hat{y}(x) = \frac{\sum_{x_i \in N_k(x)} s(x, x_i) y_i}{\sum_{x_i \in N_k(x)} s(x, x_i)}$$

3.1 k近邻算法

非参数KNN

- 没有参数需要学习
  - 实际上有N个参数：每个实例都是一个参数
  - 有N/k个有效参数
    - 直觉上讲，如果邻域不重叠，就会有N/k个邻域，每个邻域都对应一个参数
  - 超参数k
    - 我们不能将训练集上的平方误差和作为选择k的标准，因为$k = 1$时结果总是最好的
    - 在验证集上调整k的取值

3.1 k近邻算法

k近邻思想在电影推荐中的应用

- 根据用户以前喜欢的电影，如何推荐更多用户想要的电影?
  - 方法1：推荐与用户喜欢的电影有共同的演员/导演/流派的电影
  - 方法2：推荐与用户有类似兴趣的用户喜欢的电影

3.1 k近邻算法

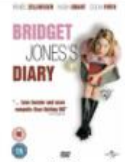
信息过滤

- 信息过滤处理用户可能感兴趣或有用的信息的传递
  - 推荐系统：以建议的形式进行信息过滤
  - 两种信息过滤方法
    - 基于内容的过滤
      - 推荐与用户喜欢的电影共享演员/导演/流派的电影
    - 协同过滤（Collaborative filtering）
      - 推荐与用户有类似兴趣的用户喜欢的电影

http://recommender-systems.org/information-filtering/

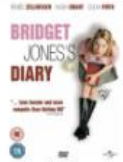
3.1 k近邻算法

一个（小的）评分矩阵

	 Die Hard	 Mission: Impossible	 GoldenEye	 Casino Royale	 Titanic	 Notting Hill	 Bridget Jones's Diary	 Love Actually
Boris	★★★☆☆	★★★★☆	★★★★★			★☆☆☆☆		
Dave		★★★★★	★★★★★	★★★★★				★☆☆☆☆
Will		★★☆☆☆			★★★★★	★★★★★	★★★☆☆	★★★★☆
George	★★★★☆	★★★★★	★★★★☆	★★★★☆				★★☆☆☆

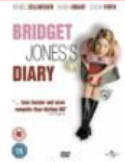
3.1 k近邻算法

用户（User）

	 Die Hard	 Mission: Impossible	 GoldenEye	 Casino Royale	 Titanic	 Notting Hill	 Bridget Jones's Diary	 Love Actually
Boris	★★★☆☆	★★★★☆	★★★★★			★☆☆☆☆		
Dave		★★★★★	★★★★★	★★★★★				★☆☆☆☆
Will		★★☆☆☆			★★★★★	★★★★★	★★★☆☆	★★★★☆
George	★★★★☆	★★★★★	★★★★☆	★★★★☆				★★☆☆☆

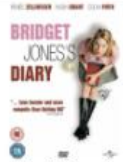
3.1 k近邻算法

物品（Item）

	 Die Hard	 Mission: Impossible	 GoldenEye	 Casino Royale	 Titanic	 Notting Hill	 Bridget Jones's Diary	 Love Actually
Boris	★★★☆☆	★★★★☆	★★★★★			★☆☆☆☆		
Dave		★★★★★	★★★★★	★★★★★				★☆☆☆☆
Will		★★☆☆☆			★★★★★	★★★★★	★★★☆☆	★★★★☆
George	★★★★☆	★★★★★	★★★★☆	★★★★☆				★★☆☆☆

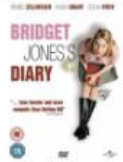
3.1 k近邻算法

用户-物品评分

	 Die Hard	 Mission: Impossible	 GoldenEye	 Casino Royale	 Titanic	 Notting Hill	 Bridget Jones's Diary	 Love Actually
Boris	★★★☆☆	★★★★☆	★★★★★			★☆☆☆☆		
Dave	★★★★★	★★★★★	★★★★★	★★★★★				★☆☆☆☆
Will		★★☆☆☆			★★★★★	★★★★★	★★★☆☆	★★★★☆
George	★★★★☆	★★★★★	★★★★☆	★★★★☆				★★☆☆☆

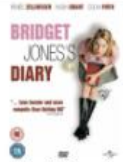
3.1 k近邻算法

用户画像

	 Die Hard	 Mission: Impossible	 GoldenEye	 Casino Royale	 Titanic	 Notting Hill	 Bridget Jones's Diary	 Love Actually
Boris	★★★☆☆	★★★★☆	★★★★★			★☆☆☆☆		
Dave		★★★★★	★★★★★	★★★★★				★☆☆☆☆
Will		★★☆☆☆			★★★★★	★★★★★	★★★☆☆	★★★★☆
George	★★★★☆	★★★★★	★★★★☆	★★★★☆				★★☆☆☆

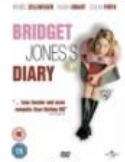
3.1 k近邻算法

物品画像

	 Die Hard	 Mission: Impossible	 GoldenEye	 Casino Royale	 Titanic	 Notting Hill	 Bridget Jones's Diary	 Love Actually
Boris	★★★☆☆	★★★★☆	★★★★★			★☆☆☆☆		
Dave		★★★★★	★★★★★	★★★★★				★☆☆☆☆
Will		★★☆☆☆			★★★★★	★★★★★	★★★☆☆	★★★★☆
George	★★★★☆	★★★★★	★★★★☆	★★★★☆				★★☆☆☆

3.1 k近邻算法

一个空的评分条目

	 Die Hard	 Mission: Impossible	 GoldenEye	 Casino Royale	 Titanic	 Notting Hill	 Bridget Jones's Diary	 Love Actually
Boris	★★★☆☆	★★★★★	★★★★★			★☆☆☆☆		
Dave		★★★★★	★★★★★	★★★★★				★☆☆☆☆
Will		★★☆☆☆			★★★★★	★★★★★	★★★☆☆	★★★★☆
George	★★★★☆	★★★★★	★★★★☆	★★★★☆				★★☆☆☆

- 在显式评分数据上的推荐
  - 预测空评分

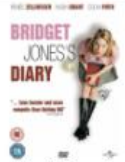
西安交通大学

XI'AN JIAOTONG UNIVERSITY

3.2 基于近邻算法的协同过滤

3.2 基于近邻算法的协同过滤

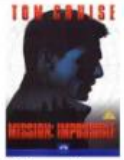
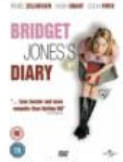
一个空的评分条目

	 Die Hard	 Mission: Impossible	 GoldenEye	 Casino Royale	 Titanic	 Notting Hill	 Bridget Jones's Diary	 Love Actually
Boris	★★★☆☆	★★★★☆	★★★★★			★☆☆☆☆		?
Dave		★★★★★	★★★★★	★★★★★				★☆☆☆☆
Will		★★☆☆☆			★★★★★	★★★★★	★★★☆☆	★★★★☆
George	★★★★☆	★★★★★	★★★★☆	★★★★☆				★★☆☆☆

- 你觉得它的评分会是多少?

3.2 基于近邻算法的协同过滤

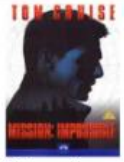
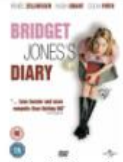
基于用户的KNN解决方法

	 Die Hard	 Mission: Impossible	 GoldenEye	 Casino Royale	 Titanic	 Notting Hill	 Bridget Jones's Diary	 Love Actually
Boris	★★★☆☆	★★★★☆	★★★★★			★☆☆☆☆		?
Dave		★★★★★	★★★★★	★★★★★				★☆☆☆☆
Will		★★☆☆☆			★★★★★	★★★★★	★★★☆☆	★★★★☆
George	★★★★☆	★★★★★	★★★★☆	★★★★☆				★★☆☆☆

- 寻找Boris的类似用户（邻居）
  - Dave和George

3.2 基于近邻算法的协同过滤

评分预测

	 Die Hard	 Mission: Impossible	 GoldenEye	 Casino Royale	 Titanic	 Notting Hill	 Bridget Jones's Diary	 Love Actually	
Boris	★★★☆☆	★★★★☆	★★★★★			★☆☆☆☆		?	
Dave		★★★★★	★★★★★	★★★★★				★☆☆☆☆	
Will		★★☆☆☆			★★★★★	★★★★★	★★★☆☆	★★★★☆	
George	★★★★☆	★★★★★	★★★★☆	★★★★☆				★★☆☆☆	

- 平均Dave和George在Love Actually上的评分
  - 预测 = (1+2)/2 = 1.5

3.2 基于近邻算法的协同过滤

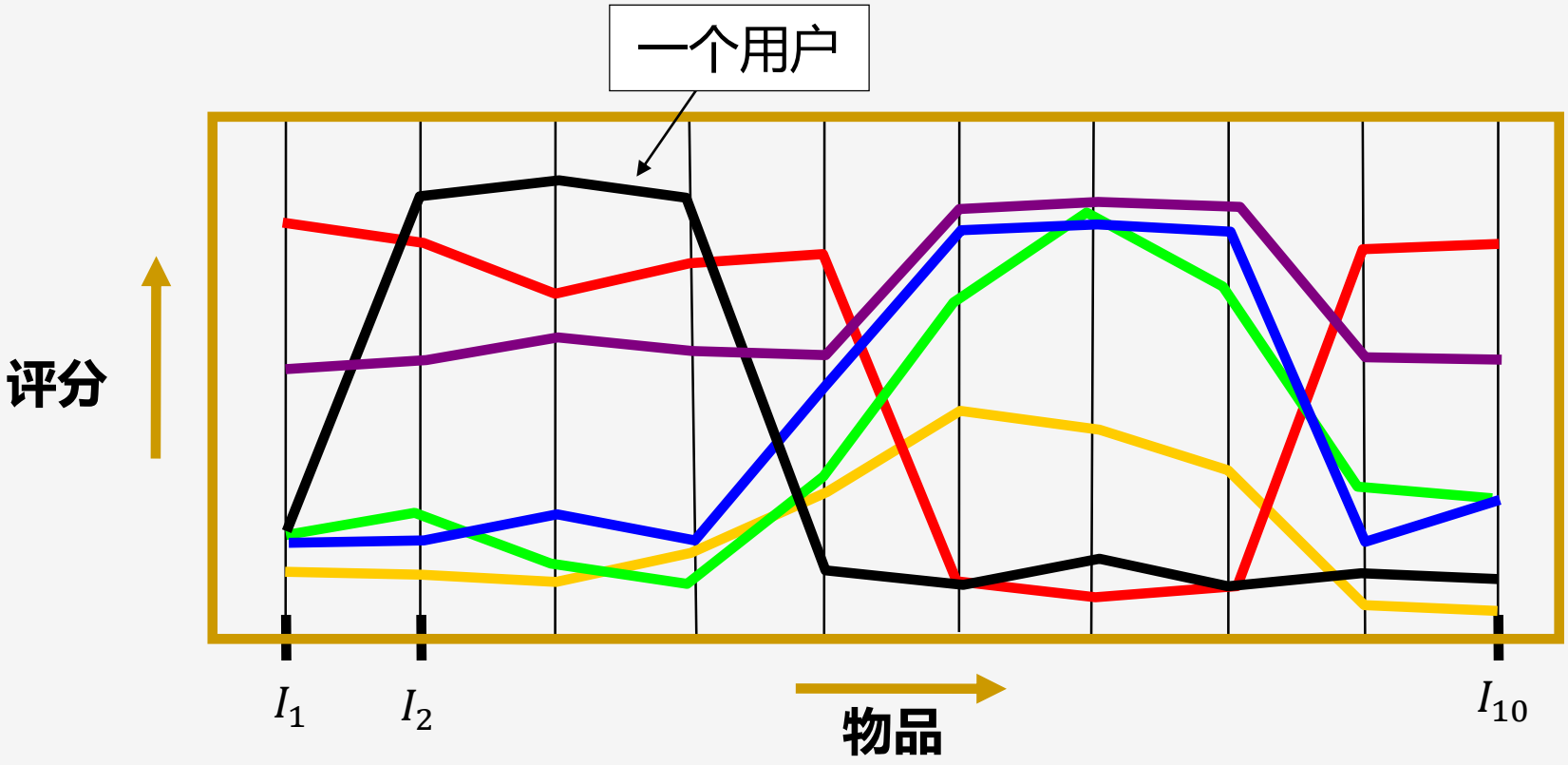
用于推荐的协同过滤

	Die Hard	Mission: Impossible	GoldenEye	Casino Royale	Titanic	Notting Hill	Bridget Jones's Diary	Love Actually
Boris	★★★☆☆	★★★★☆	★★★★★			★☆☆☆☆		?
Dave		★★★★★	★★★★★	★★★★★				★☆☆☆☆
Will		★★☆☆☆			★★★★★	★★★★★	★★★☆☆	★★★★☆
George	★★★★☆	★★★★★	★★★★☆	★★★★☆				★★☆☆☆

- 基于用户的基本KNN算法
  - 为每个目标用户提供推荐
    - 寻找类似用户
    - 基于类似用户推荐新物品

3.2 基于近邻算法的协同过滤

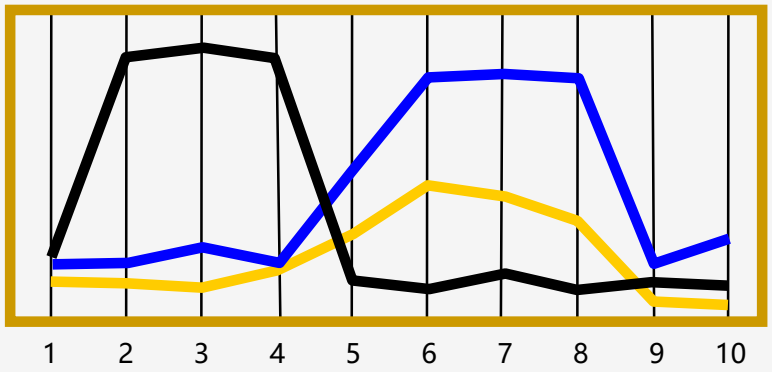
用户的相似度

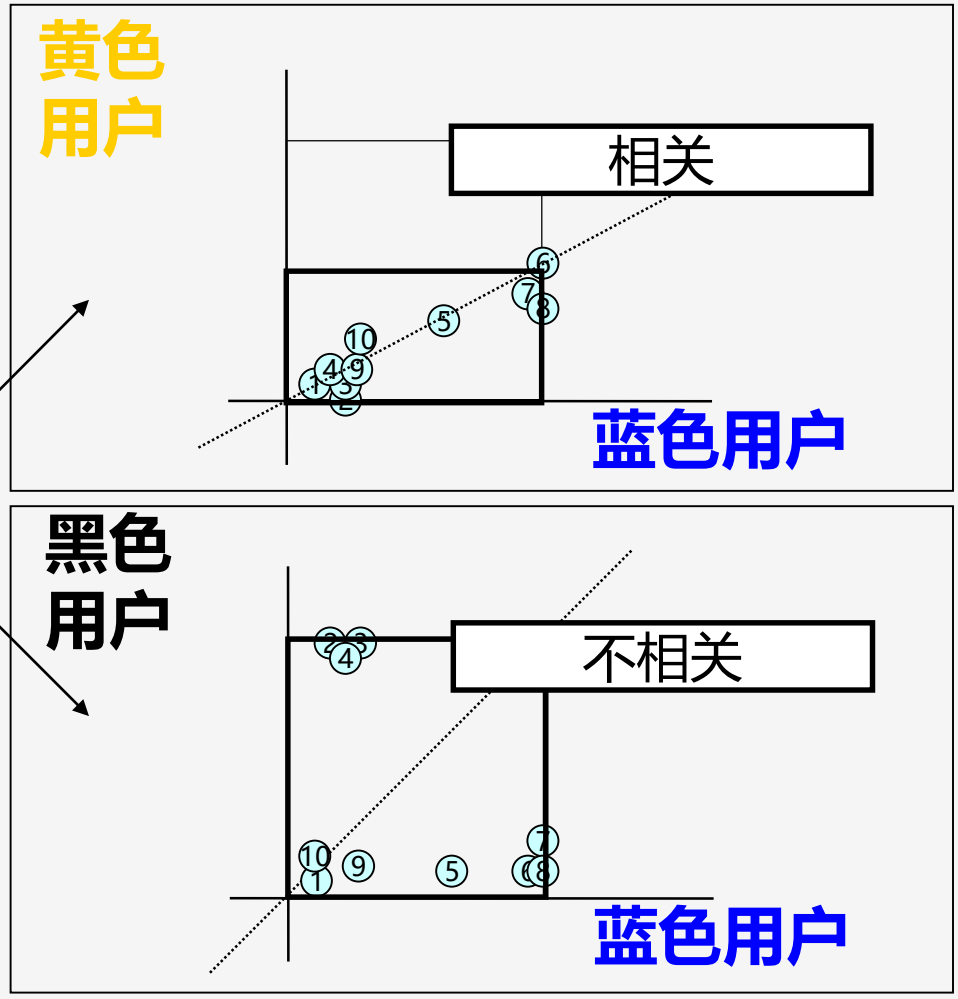


- 每个用户的画像可以根据其选择的物品评级直接构建为向量

3.2 基于近邻算法的协同过滤

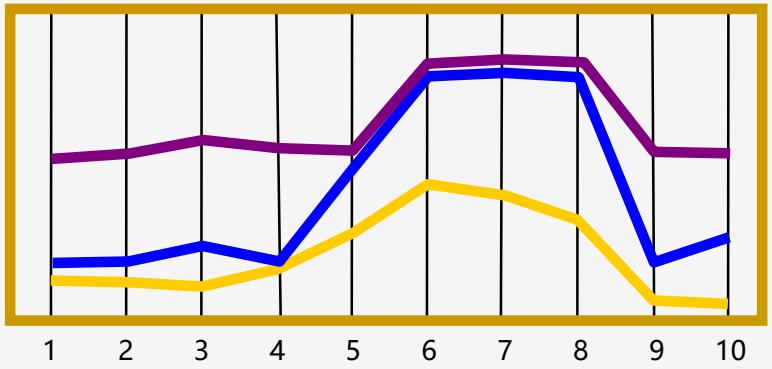
用户的相似度

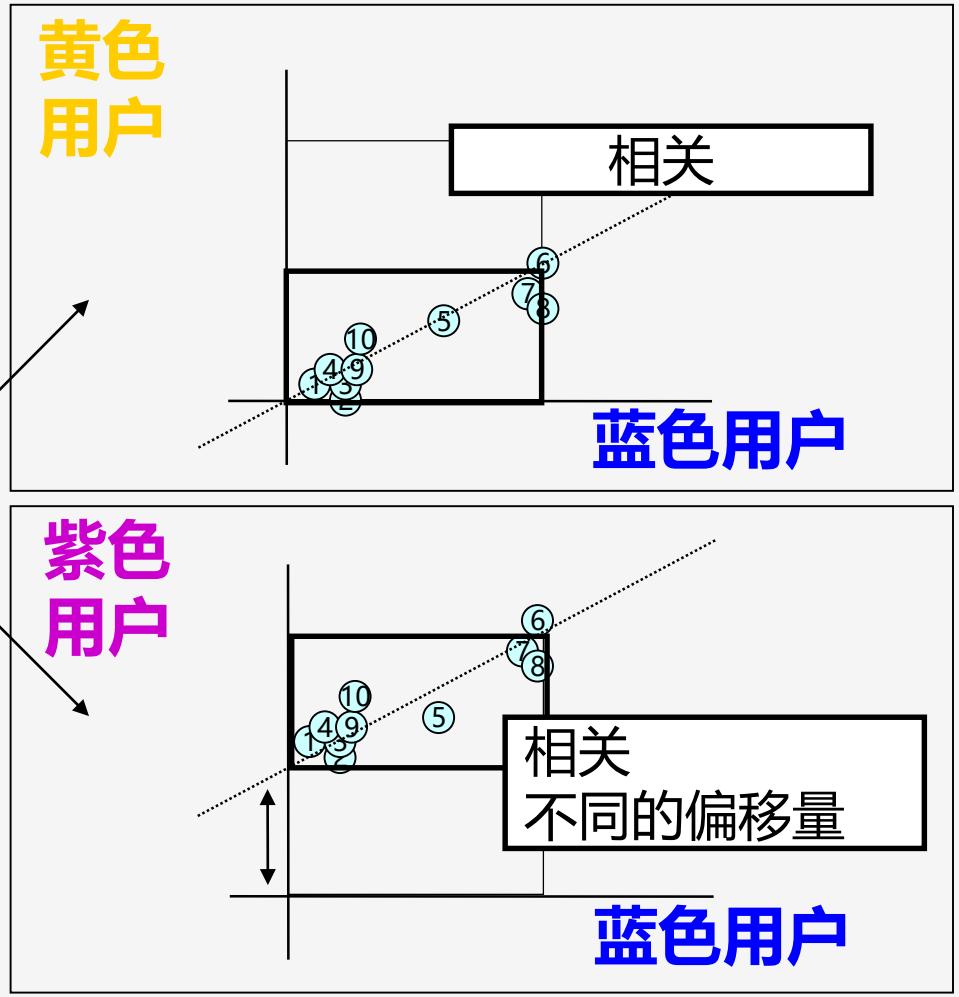




3.2 基于近邻算法的协同过滤

用户的相似度





用户的相似度

- 两个用户a和b之间的相似度量
  - 余弦（角）相似度

$$s_u^{cos}(u_a, u_b) = \frac{u_a^{\top} u_b}{\| u_a \| \| u_b \|} = \frac{\sum_m x_{a,m} x_{b,m}}{\sqrt{\sum_m x_{a,m}^2 \sum_m x_{b,m}^2}}$$

  - 皮尔森相关性

$$s_u^{corr}(u_a, u_b) = \frac{\sum_m (x_{a,m} - \overline{x}_a)(x_{b,m} - \overline{x}_b)}{\sqrt{\sum_m (x_{a,m} - \overline{x}_a)^2 \sum_m (x_{b,m} - \overline{x}_b)^2}}$$

3.2 基于近邻算法的协同过滤

基于用户的KNN评分预测

- 预测从目标用户t与物品m之间的评分

根据和目标用户的相似度定义的权重

根据相似用户的评分进行预测（针对该用户的平均评分进行校正）

$$\hat{x}_{t,m} = \overline{x}_t + \frac{\sum_{u_a \in N_u(u_t)} s_u(u_t, u_a)(x_{a,m} - \overline{x}_a)}{\sum_{u_a \in N_u(u_t)} s_u(u_t, u_a)}$$

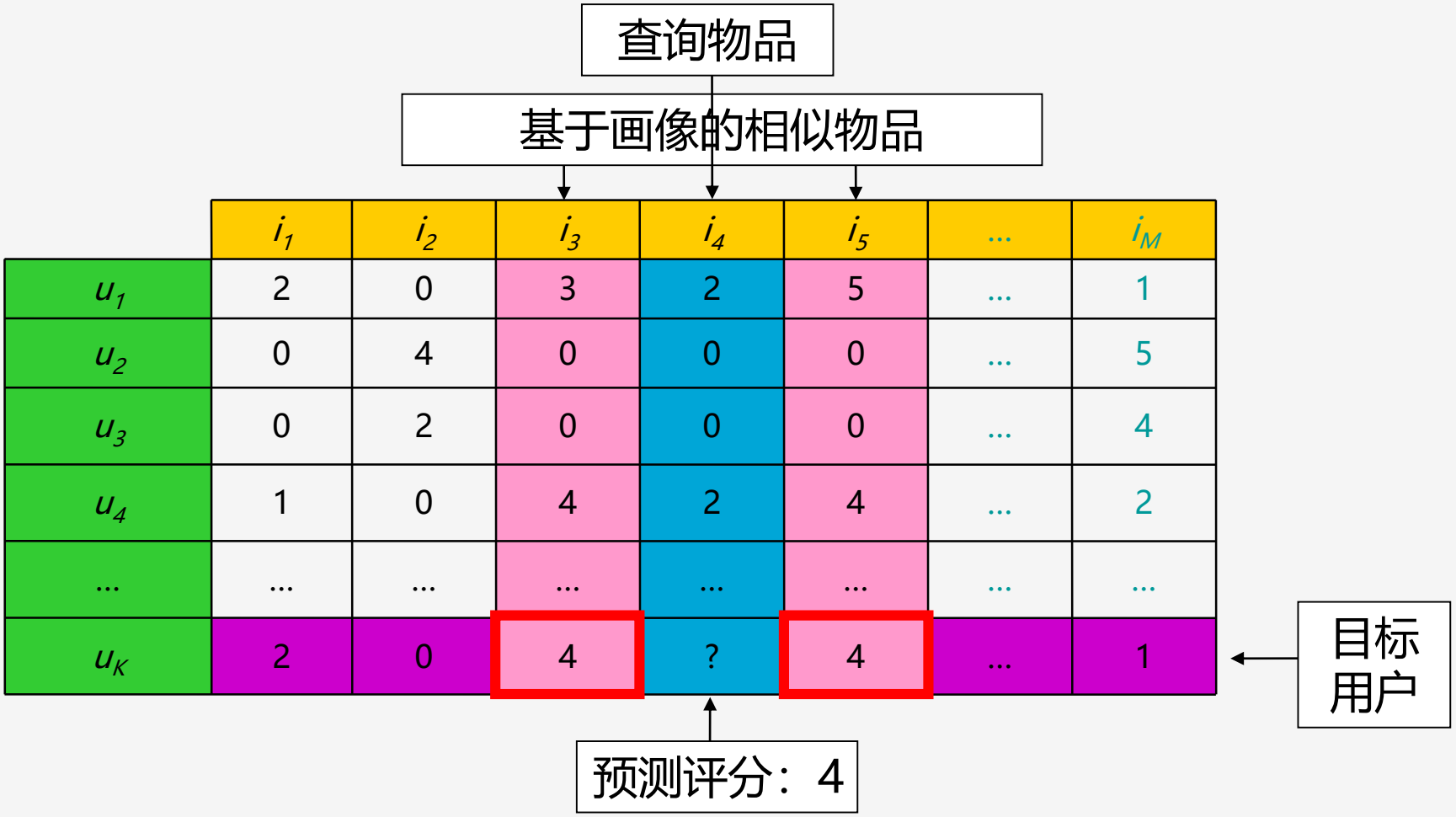
对目标用户的平均评分的修正

$$\overline{x}_t = \frac{1}{|I(u_t)|} \sum_{m \in I(u_t)} x_{t,m}$$

标准化，使评分在合理的范围内

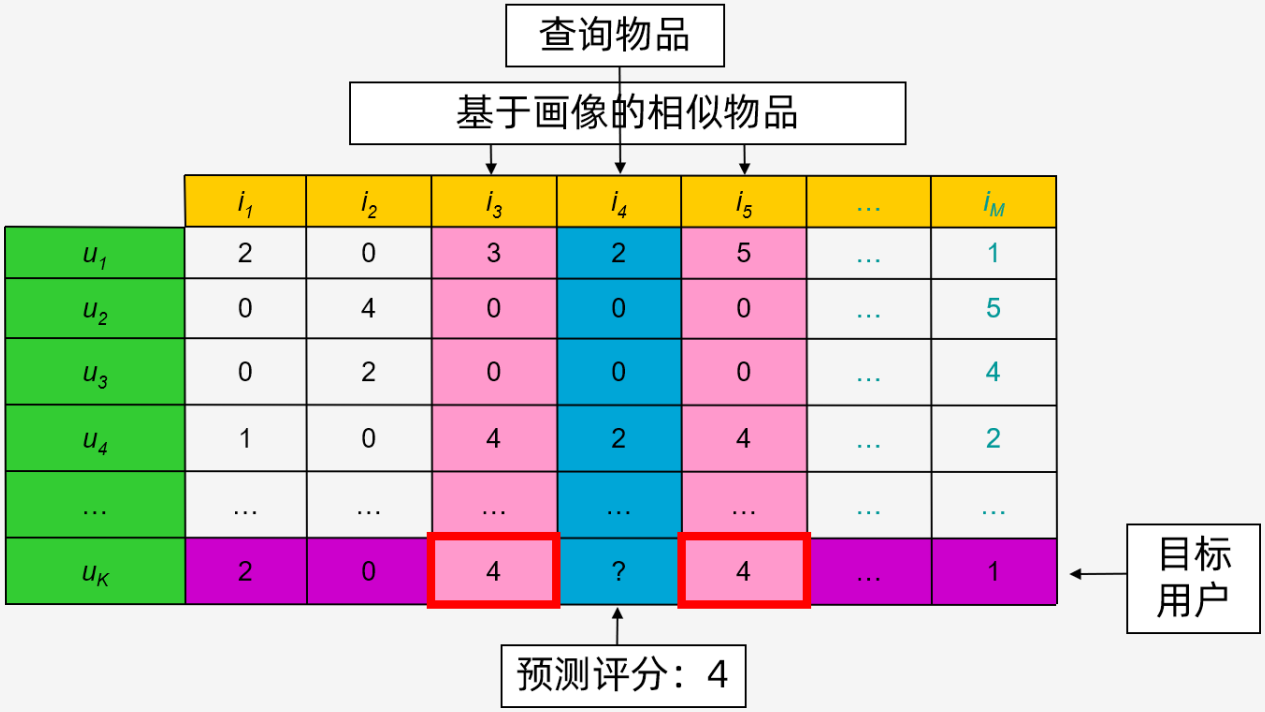
基于物品的KNN评分预测

- 基于物品相似度的推荐



基于物品的KNN评分预测

- 对于目标用户t的每个未评分物品m
  - 寻找类似物品{a}
  - 基于类似物品的集合{a}预测物品m的评分



物品的相似度

- 两个物品a和b之间的相似度量
  - 余弦（角）相似度

$$s_i^{cos}(i_a, i_b) = \frac{i_a^{\top} i_b}{\| i_a \| \| i_b \|} = \frac{\sum_u x_{u,a} x_{u,b}}{\sqrt{\sum_u x_{u,a}^2 \sum_u x_{u,b}^2}}$$

  - 调整后的余弦相似度（$\overline{x}_u$ 为第u个用户对所有物品的平均评分）

$$s_i^{adcos}(i_a, i_b) = \frac{\sum_u (x_{u,a} - \overline{x}_u)(x_{u,b} - \overline{x}_u)}{\sqrt{\sum_u (x_{u,a} - \overline{x}_u)^2 \sum_u (x_{u,b} - \overline{x}_u)^2}}$$

  - 皮尔森相关性（$\overline{x}_a$、 $\overline{x}_b$分别为所有用户对物品a和物品b的平均评分）

$$s_i^{corr}(i_a, i_b) = \frac{\sum_u (x_{u,a} - \overline{x}_a)(x_{u,b} - \overline{x}_b)}{\sqrt{\sum_u (x_{u,a} - \overline{x}_a)^2 \sum_u (x_{u,b} - \overline{x}_b)^2}}$$

Badrul Sarwar, George Karypis, Joseph Konstan, and John Riedl. “Item-Based Collaborative Filtering Recommendation Algorithms” WWW 2010.

基于物品的KNN评分预测

- 获取目标用户t评分过，且和物品i_a最相似的k个最近邻物品

排序位置

$$N_i(u_t, i_a) = \{i_b | r_i(i_a, i_b) < K^*, x_{t,b} \neq 0\}$$

选择K^*，使$|N_i(u_t, i_a)| = k$

- 预测目标用户未评分的物品i_a的评分

$$\hat{x}_{t,a} = \frac{\sum_{i_b \in N_i(u_t, i_a)} s_i(i_a, i_b) x_{t,b}}{\sum_{i_b \in N_i(u_t, i_a)} s_i(i_a, i_b)}$$

由于使用目标用户本身数据进行预测，因此无需修正用户的平均评分

实证研究

- Movielens数据集来自
  - https://grouplens.org/datasets/movielens/
  - 用户访问Movielens
    - 评分和收到电影的推荐
  - 数据集（ML-100k）
  - 100k个从1到5的评分
  - 943位用户，1682部电影（至少有一位用户评分）
  - 稀疏程度

$$1 - \frac{\#\text{non-zero entries}}{\text{total entries}} = 1 - \frac{10^5}{943 \times 1682} = 93.69\%$$

Badrul Sarwar, George Karypis, Joseph Konstan, and John Riedl. “Item-Based Collaborative Filtering Recommendation Algorithms” WWW 2010.

实验设置

- 把数据拆分成训练集($x\%$)和测试集($(100-x)\%$)
  - 可重复T次，结果取平均值
- 评估指标
  - 平均绝对误差（MAE）

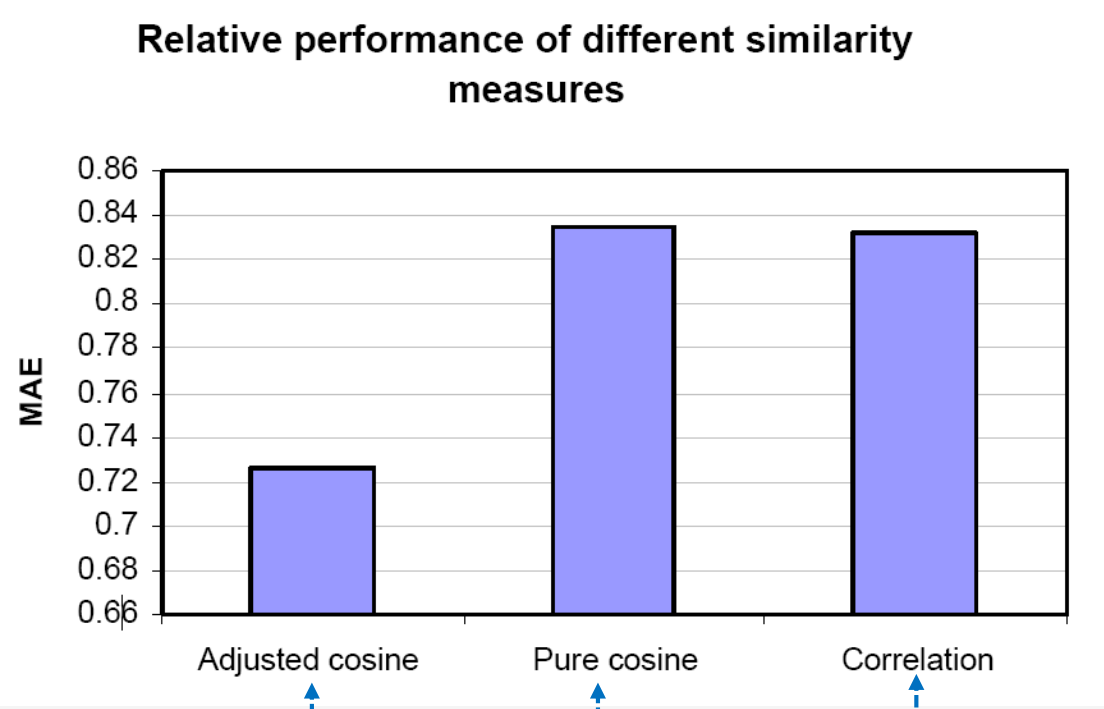
$$\text{MAE} = \frac{1}{|D_{\text{test}}|} \sum_{(u,i,r)\in D_{\text{test}}} |r - \hat{r}_{u,i}|$$

  - 均方根误差（RMSE）

$$\text{RMSE} = \sqrt{\frac{1}{|D_{\text{test}}|} \sum_{(u,i,r)\in D_{\text{test}}} \left(r - \hat{r}_{u,i}\right)^2}$$

3.2 基于近邻算法的协同过滤

相似度量的表现



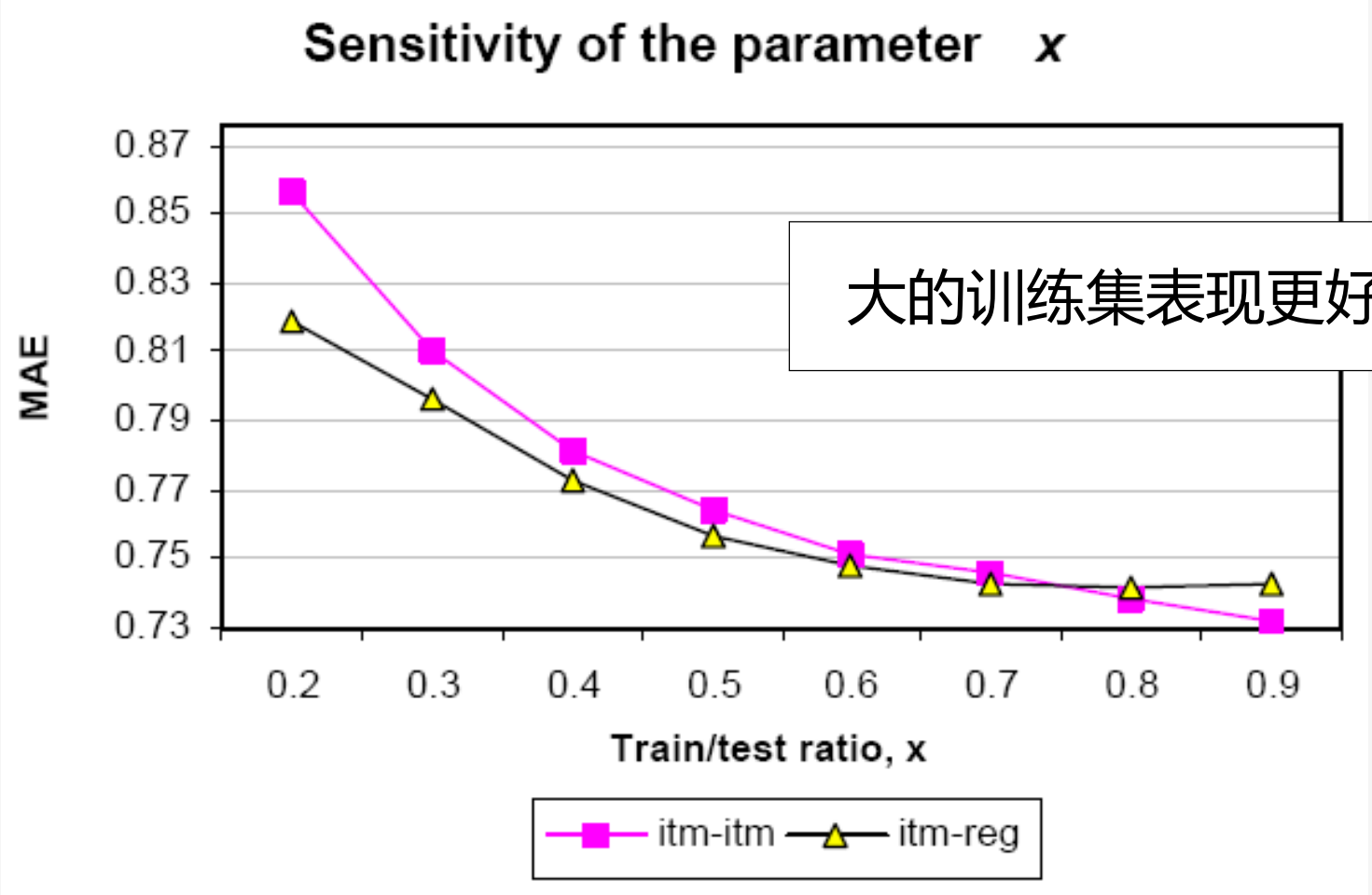
$$s_i^{adcos}(i_a, i_b) = \frac{\sum_u (x_{u,a} - \overline{x}_u)(x_{u,b} - \overline{x}_u)}{\sqrt{\sum_u (x_{u,a} - \overline{x}_u)^2 \sum_u (x_{u,b} - \overline{x}_u)^2}}$$

$$s_i^{corr}(i_a, i_b) = \frac{\sum_u (x_{u,a} - \overline{x}_a)(x_{u,b} - \overline{x}_b)}{\sqrt{\sum_u (x_{u,a} - \overline{x}_a)^2 \sum_u (x_{u,b} - \overline{x}_b)^2}}$$

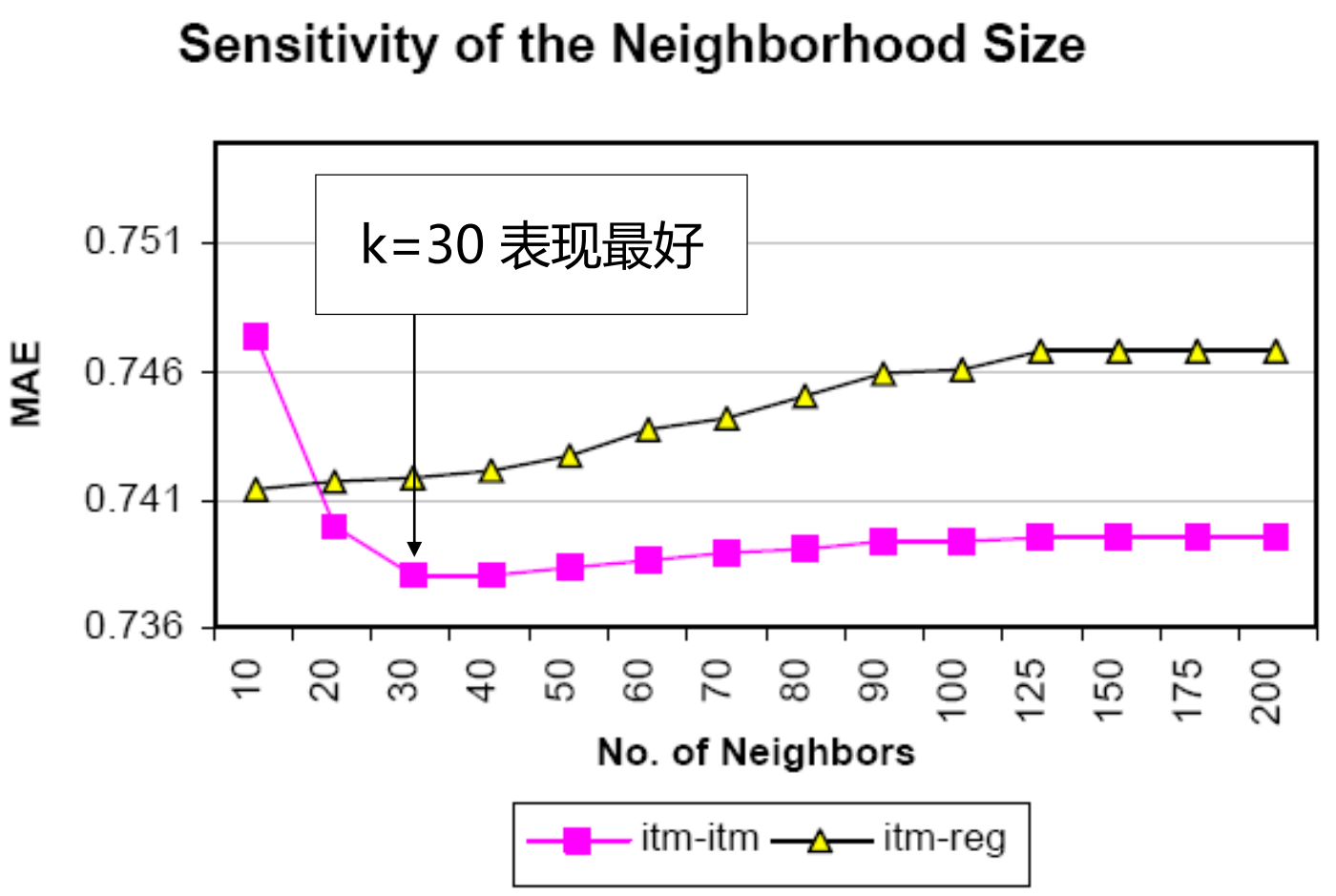
$$s_i^{cos}(i_a, i_b) = \frac{\sum_u x_{u,a} x_{u,b}}{\sqrt{\sum_u x_{u,a}^2 \sum_u x_{u,b}^2}}$$

3.2 基于近邻算法的协同过滤

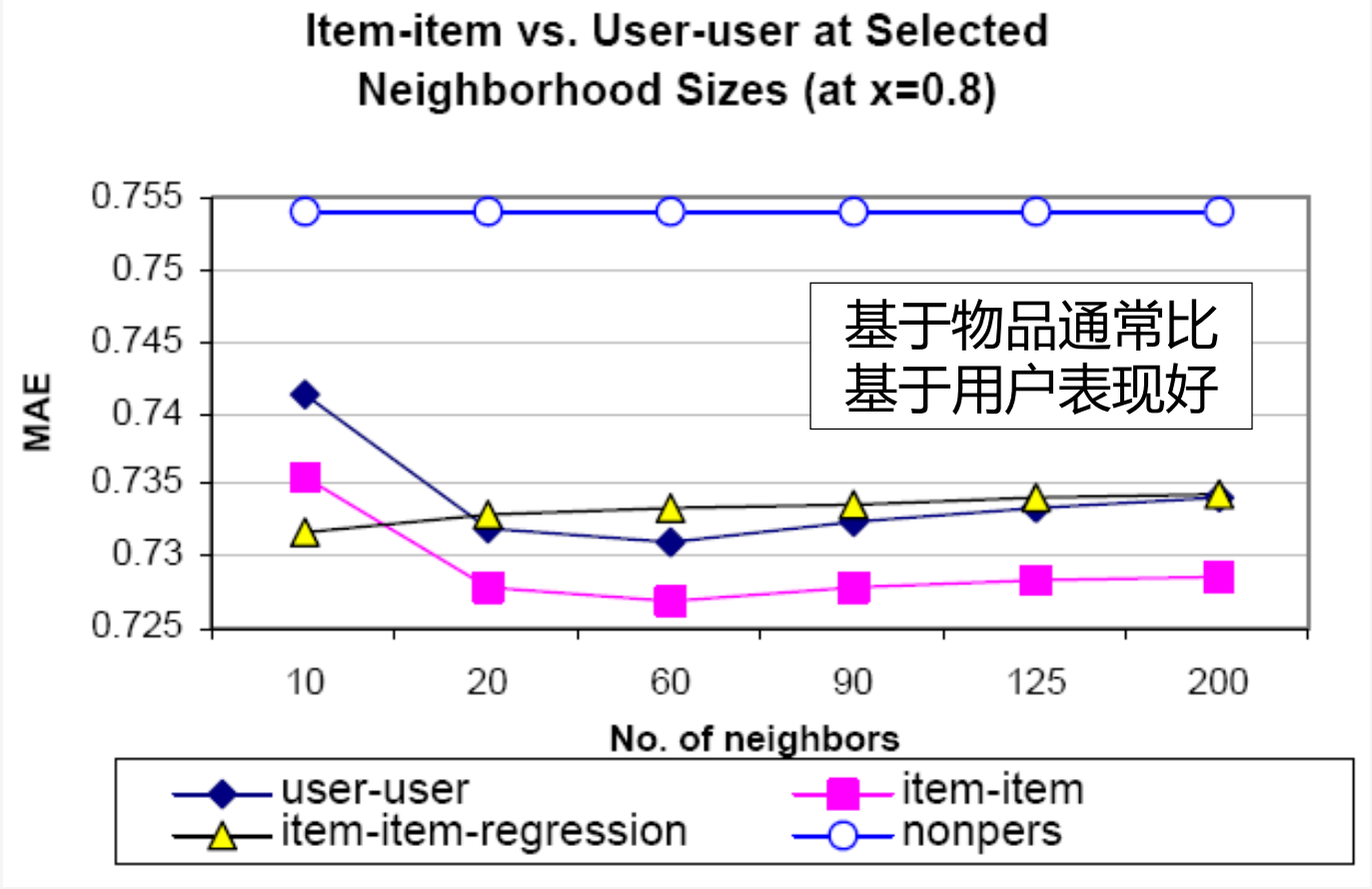
训练/测试比例的敏感度



邻居大小k的敏感度


Sensitivity of the Neighborhood Size
k=30 表现最好
MAE
0.751
0.746
0.741
0.736
10
20
30
40
50
60
70
80
90
100
125
150
175
200
No. of Neighbors
itm-itm
itm-reg

基于物品 vs. 基于用户



- 物品-物品相似度通常更加稳定和客观

3.2 基于近邻算法的协同过滤

Item-Based Collaborative Filtering Recommendation Algorithms

Badrul Sarwar, George Karypis, Joseph Konstan, and John Riedl
{sarwar, karypis, konstan, riedl}@cs.umn.edu
GroupLens Research Group/Army HPC Research Center
Department of Computer Science and Engineering
University of Minnesota, Minneapolis, MN 55455

ABSTRACT

Recommender systems apply knowledge discovery techniques to the problem of making personalized recommendations for information, products or services during a live interaction. These systems, especially the k-nearest neighbor collaborative filtering based ones, are achieving widespread success on the Web. The tremendous growth in the amount of available information and the number of visitors to Web sites in recent years poses some key challenges for recommender systems. These are: producing high quality recommendations, performing many recommendations per second for millions of users and items and achieving high coverage in the face of data sparsity. In traditional collaborative filtering systems the amount of work increases with the number of participants in the system. New recommender system technologies are needed that can quickly produce high quality recommendations, even for very large-scale problems. To address these issues we have explored item-based collaborative filtering techniques. Item-based techniques first analyze the user-item matrix to identify relationships between different items, and then use these relationships to indirectly compute recommendations for users.

In this paper we analyze different item-based recommendation generation algorithms. We look into different techniques for computing item-item similarities (e.g., item-item correlation vs. cosine similarities between item vectors) and different techniques for obtaining recommendations from them (e.g., weighted sum vs. regression model). Finally, we experimentally evaluate our results and compare them to the basic k-nearest neighbor approach. Our experiments suggest that item-based algorithms provide dramatically better performance than user-based algorithms, while at the same time providing better quality than the best available user-based algorithms.

through all the available information to find that which is most valuable to us.

One of the most promising such technologies is *collaborative filtering* [19, 27, 14, 16]. Collaborative filtering works by building a database of preferences for items by users. A new user, Neo, is matched against the database to discover *neighbors*, which are other users who have historically had similar taste to Neo. Items that the neighbors like are then recommended to Neo, as he will probably also like them. Collaborative filtering has been very successful in both research and practice, and in both information filtering applications and E-commerce applications. However, there remain important research questions in overcoming two fundamental challenges for collaborative filtering recommender systems.

The first challenge is to improve the scalability of the collaborative filtering algorithms. These algorithms are able to search tens of thousands of potential neighbors in real-time, but the demands of modern systems are to search tens of millions of potential neighbors. Further, existing algorithms have performance problems with individual users for whom the site has large amounts of information. For instance, if a site is using browsing patterns as indications of content preference, it may have thousands of data points for its most frequent visitors. These “long user rows” slow down the number of neighbors that can be searched per second, further reducing scalability.

The second challenge is to improve the quality of the recommendations for the users. Users need recommendations they can trust to help them find items they will like. Users will ”vote with their feet” by refusing to use recommender systems that are not consistently accurate for them.

In some ways these two challenges are in conflict, since the less time an algorithm spends searching for neighbors, the more scalable it will be, and the worse its quality. For this reason, it is important to treat the two challenges simul-

基于KNN的方法总结

- 直接，有高度可解释性
- 没有参数学习
  - 只有一个超参数k可以调整
  - 无法通过学习得到改善
- 效率可能是一个严重的问题
  - 当用户/项目编号很大时
  - 当有大量的用户-物品评分时
- 我们可能需要一个参数化和可学习的模型

西安交通大学

XI'AN JIAOTONG UNIVERSITY

3.3 矩阵分解

矩阵分解（Matrix Factorization）

	Die Hard	Mission: Impossible	GoldenEye	Casino Royale	Titanic	Notting Hill	Bridget Jones's Diary	Love Actually
Boris	★★★☆☆	★★★★☆	★★★★★			★☆☆☆☆		★★☆☆☆
Dave		★★★★★	★★★★★	★★★★★				★☆☆☆☆
Will		★★☆☆☆			★★★★★	★★★★★	★★★☆☆	★★★★☆
George	★★★★☆	★★★★★	★★★★☆	★★★★☆				★★☆☆☆

≅

	Die Hard	Mission: Impossible	GoldenEye	Casino Royale	Titanic	Notting Hill	Bridget Jones's Diary	Love Actually
Boris								
Dave								
Will								
George								

·

Koren, Yehuda, Robert Bell, and Chris Volinsky. "Matrix factorization techniques for recommender systems." Computer 42.8 (2009).

3.3 矩阵分解

矩阵分解（Matrix Factorization）

物品

R　　用户

1		3			5			5		4	
		5	4	?		4			2	1	3
2	4		1	2		3		4	3	5	
	2	4		5			4			2	
		4	3	4	2					2	5
1		3		3			2			4	

$$\hat{r}_{u,i} = p_u^{\top} q_i$$

i　　物品

$\approx$　　u

用户

.1	-.4	.2
-.5	.6	.5
-.2	.3	.5
1.1	2.1	.3
-.7	2.1	-2
-1	.7	.3

P

$\cdot$

1.1	-.2	.3	.5	-2	-.5	.8	-.4	.3	1.4	2.4	-.9
-.8	.7	.5	1.4	.3	-1	1.4	2.9	-.7	1.2	-.1	1.3
2.1	-.4	.6	1.7	2.4	.9	-.3	.4	.8	.7	-.6	.1

Q

3.3 矩阵分解

基本的矩阵分解模型

- 用户u在物品i上的预测

$$\hat{r}_{u,i} = p_u^\top q_i$$

- 损失函数

$$\mathcal{L}(u, i, r_{u,i}) = \frac{1}{2}\left(r_{u,i} - p_u^\top q_i\right)^2$$

- 训练目标

$$\min_{P,Q} \sum_{r_{u,i} \in R} \frac{1}{2}(r_{u,i} - p_u^\top q_i)^2 + \frac{\lambda}{2}(\| p_u \|^2 + \| q_i \|^2)$$

- 梯度

$$\frac{\partial \mathcal{L}(u, i, r_{u,i})}{\partial p_u} = (p_u^\top q_i - r_{u,i}) q_i + \lambda p_u$$

$$\frac{\partial \mathcal{L}(u, i, r_{u,i})}{\partial q_i} = (p_u^\top q_i - r_{u,i}) p_u + \lambda q_i$$

3.3 矩阵分解

带偏置的矩阵分解模型

- 用户u在物品i上的预测

$$\hat{r}_{u,i} = \mu + b_u + b_i + p_u^{\top} q_i$$

全局偏置　用户偏置　物品偏置　用户-物品交互

- 训练目标

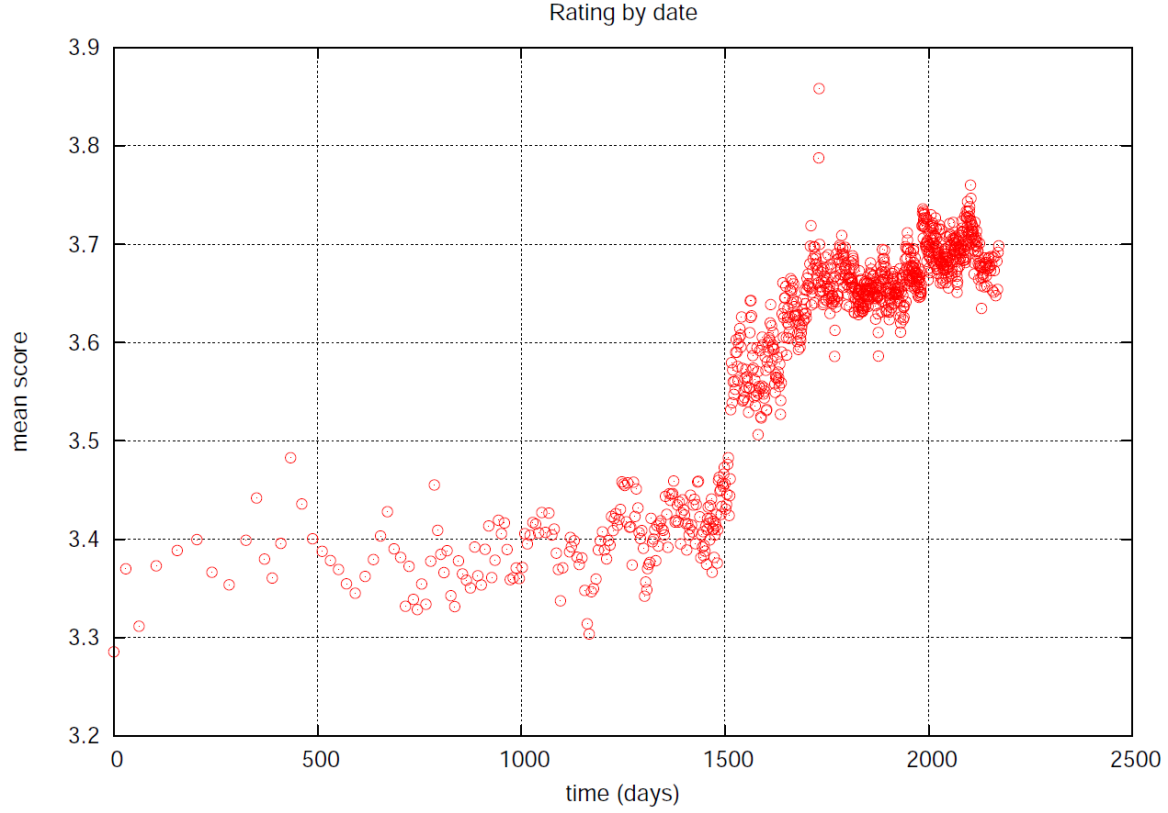
$$\min_{P,Q} \sum_{r_{u,i} \in D} \frac{1}{2}(r_{u,i} - (\mu + b_u + b_i + p_u^{\top} q_i))^2 + \frac{\lambda}{2}(\| p_u \|^2 + \| q_i \|^2 + b_u^2 + b_i^2)$$

- 梯度更新

$$\delta = r_{u,i} - (\mu + b_u + b_i + p_u^{\top} q_i)$$
$$\mu \leftarrow \mu + \eta\delta$$
$$b_u \leftarrow (1 - \eta\lambda) b_u + \eta\delta$$
$$b_i \leftarrow (1 - \eta\lambda) b_i + \eta\delta$$
$$p_u \leftarrow (1 - \eta\lambda) p_u + \eta\delta q_i$$
$$q_i \leftarrow (1 - \eta\lambda) q_i + \eta\delta p_u$$

3.3 矩阵分解

时间动态变化

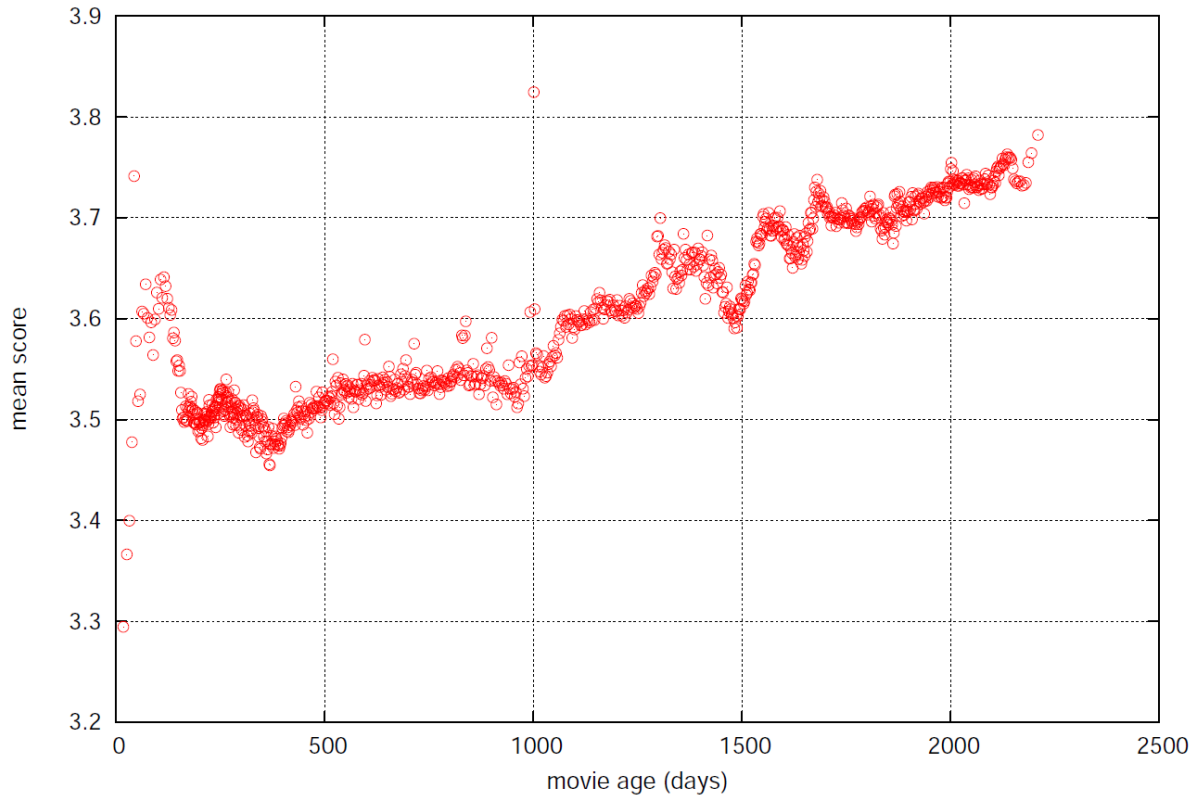


- 约1500天（2004年初）进入数据集的平均电影评分突然上升

Koren, Yehuda. "Collaborative filtering with temporal dynamics. " KDD 2009.

3.3 矩阵分解

时间动态变化



- 对于更老的电影，人们往往给予更高的评分

Koren, Yehuda. "Collaborative filtering with temporal dynamics. " KDD 2009.

动态变化的多个原因

- 物品方面的影响
  - 物品受欢迎程度不断变化
  - 季节性原因影响物品的受欢迎程度
- 用户方面的影响
  - 用户改变了他们的品味
  - 暂时的，短期偏置
  - 浮动的评分等级
  - 在一个家庭里实际评分人的改变

3.3 矩阵分解

解决动态变化问题

- 因子模型方便地允许分别处理不同原因
- 我们观察到的变化：
  - 个人用户的评分等级$b_u(t)$
  - 个人物品的受欢迎程度$b_i(t)$
  - 用户偏好$p_u(t)$

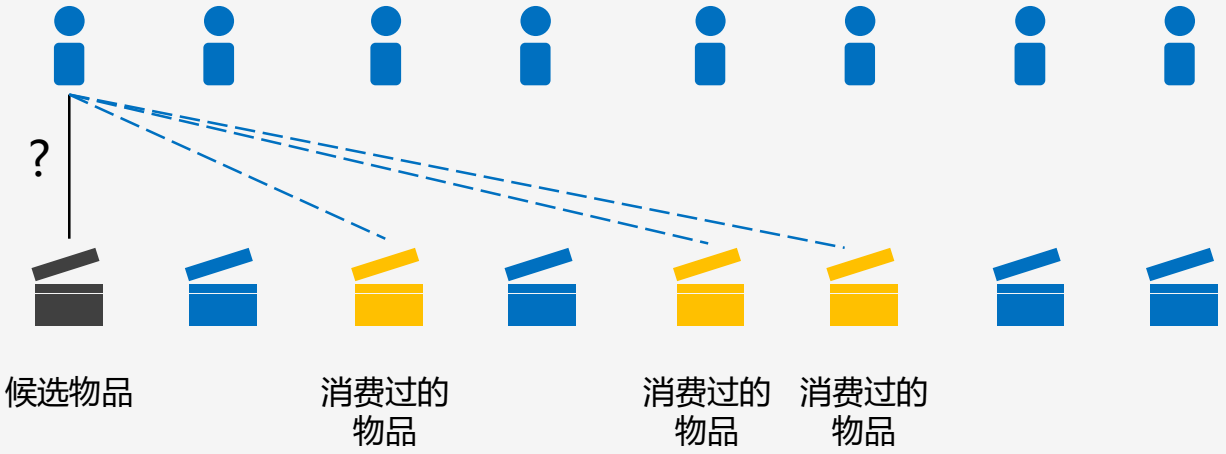
$$r_{u,i}(t) = \mu + b_u(t) + b_i(t) + p_u(t)^{\mathsf{T}} q_i$$

- 设计指南
  - 物品显示较慢的时间变化
  - 用户表现出频繁和突然的变化
  - 因子$p_u(t)$的建模成本很高
  - 通过大量参数化函数来获得灵活性

3.3 矩阵分解

基于邻居（相似性）的矩阵分解

- 假设：用户以前消费过的物品反映了她的品味
- 从这些“类似的”物品中获取未知评分（物品-物品变体）



3.3 矩阵分解

基于邻居的矩阵分解：SVD++

$$\hat{r}_{u,i} = b_{u,i} + q_i^{\top}(p_u + |N(u)|^{-\frac{1}{2}} \sum_{j \in N(u)} y_j)$$

- $\hat{r}_{u,i}$：需要估计用户u的评分物品i
- $b_{u,i}$：偏置
- q_i：物品向量
- p_u：用户向量
- $N(u)$：用户之前评分的物品的集合
- y_j：为了表示邻居相似度的物品向量

- 每个物品有两个潜在向量
  - 本身的物品向量q_i
  - 用于估计候选物品和目标用户之间的相似度的向量y_i

Koren, Yehuda. "Factorization meets the neighborhood: a multifaceted collaborative filtering model." KDD, 2008.

3.3 矩阵分解

Netflix奖

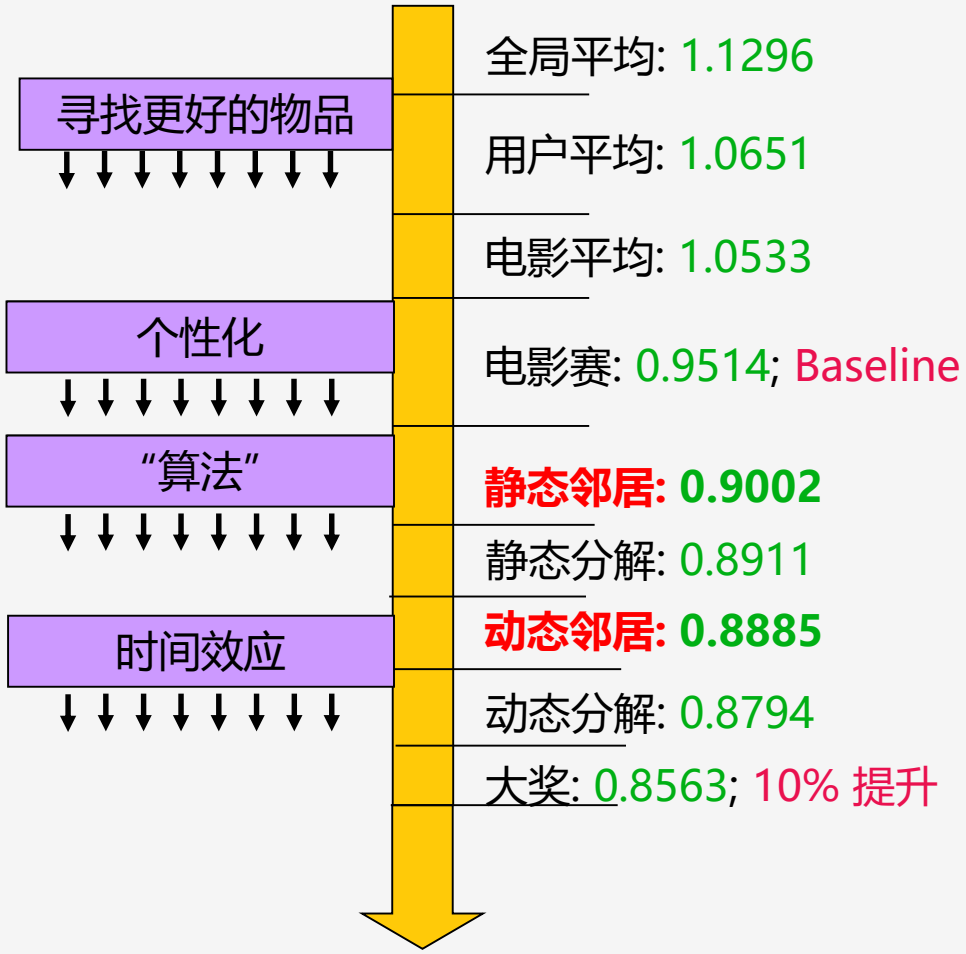
- 针对电影的最佳协同过滤算法的公开比赛
  - 始于2006年10月2日
  - 一项价值数百万美元的挑战，将Netflix推荐算法的准确度（RMSE）提高10%

- Netflix提供
  - 训练数据：100,480,507评分：
  - 480,189位用户x 17,770部电影
  - 格式：<用户，电影，日期，评级>

- 两种流行的方法：
  - 矩阵分解
  - K近邻算法

3.3 矩阵分解

Netflix奖



动态邻居模型作为动态因子模型提供相同的相对RMSE提升(0.0117)


NETFLIX
2009
DATE 09.21.09
PAY TO THE ORDER OF BellKor's Pragmatic Chaos
$ 1,000,000 00
AMOUNT ONE MILLION 00/100
FOR The Netflix Prize
Reed Hastings
The Netflix Prize

西安交通大学

XI'AN JIAOTONG UNIVERSITY

总结

k-近邻算法总结

- 协同过滤是推荐系统中的重要机器学习技术，主要探索学习数据点之间的相似性，进而对目标标签做出预测
- KNN算法是最简单的非参数化模型
- 矩阵分解是最基础的双线性模型。参数向量的内积操作是双线性模型中最常使用的操作，这时对一边参数向量求梯度，得到另一边参数向量的取值
- 因子分解机是一种通过向量内积来直接学习两两特征交互对标签预测的影响的通用模型

西安交通大学
XI'AN JIAOTONG UNIVERSITY

谢谢大家！