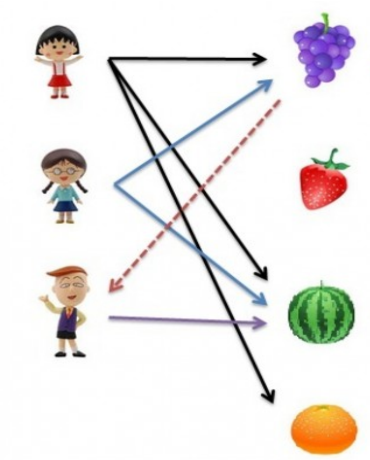
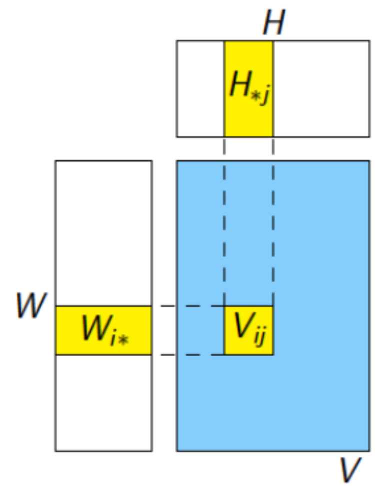
# Tufts COMP 135: Introduction to Machine Learning

https://www.cs.tufts.edu/comp/135/2019s/

# Recommendation Systems



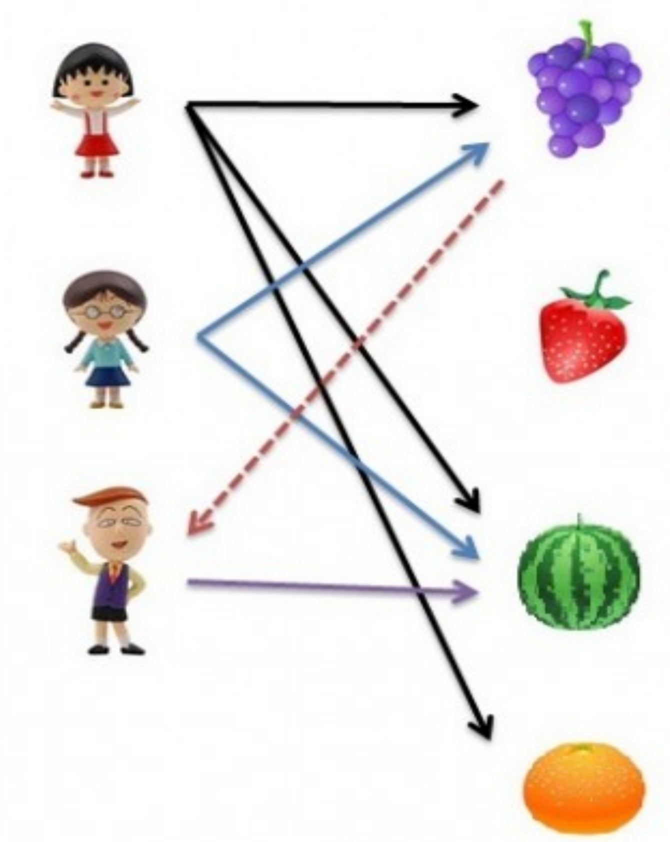
| | | | |
|---|---|---|---|
| 2 | | 4 | 1 |
| 5 | | 3 | |
| 2 | 4 | 5 | |

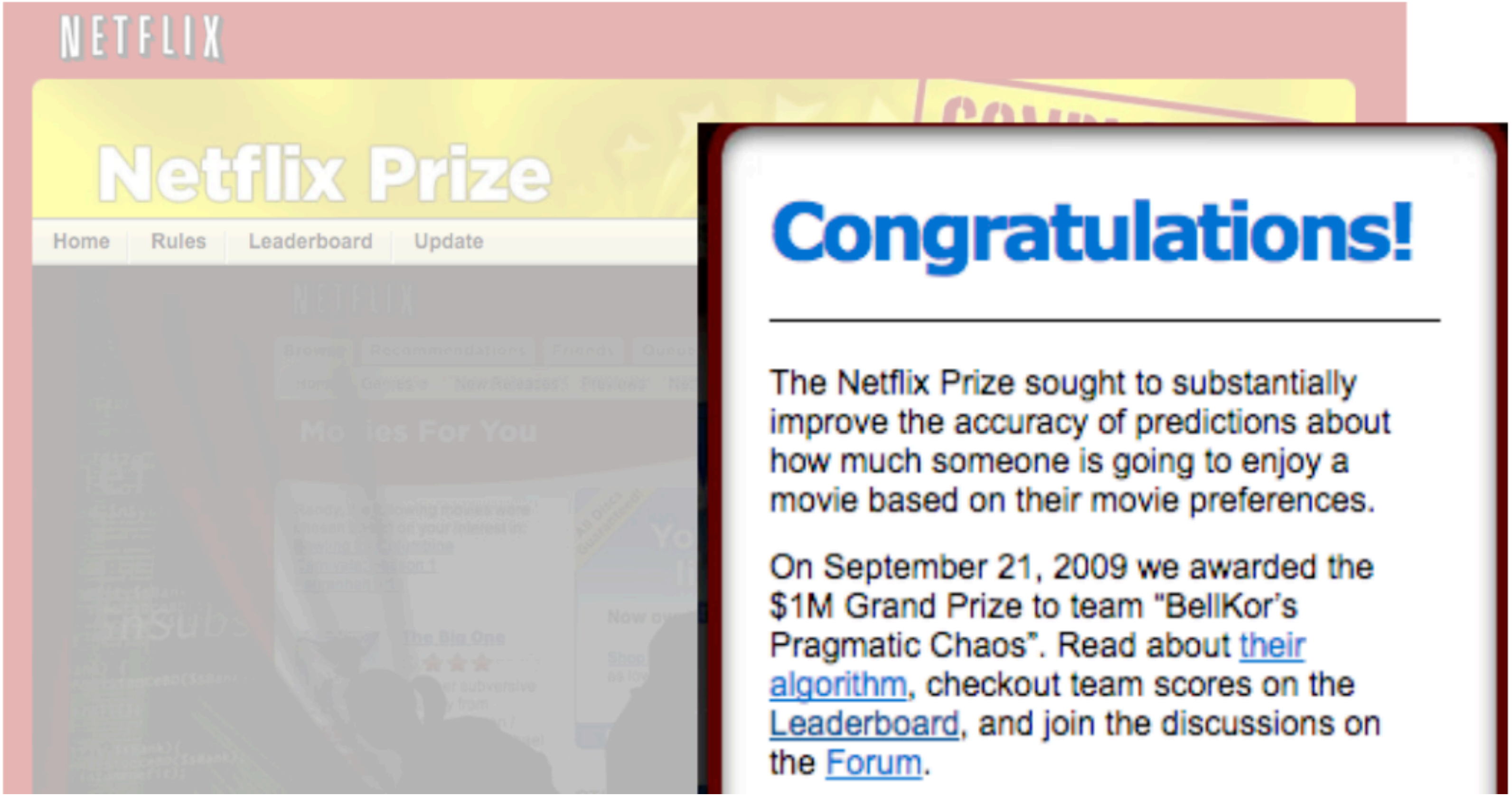

Prof. Mike Hughes

*Many ideas/slides attributable to:*
*Liping Liu (Tufts), Emily Fox (UW)*
*Matt Gormley (CMU)*

# Recommendation Task: Which users will like which items?



# Motivation: Netflix Prize



# Recommendation Systems Objectives (day 24)

- Explain two major types of recommendation
  - Content-based filtering
    - Supervised learning problem where
      - Each item has known features
      - Each user has known features
  - Collaborative filtering
    - Unsupervised learning problem
    - Approach 1: Neighbor-based recommendation
    - Approach 2: Latent Factor Methods

# Task: Recommendation

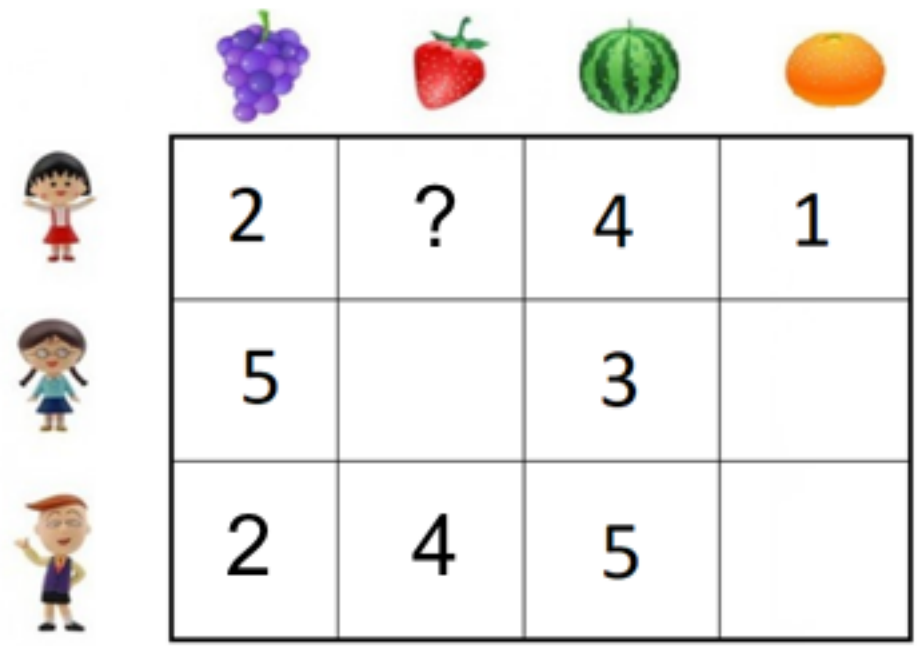
**Supervised Learning**

**Content filtering**

**Unsupervised Learning**

**Collaborative filtering**

Reinforcement Learning



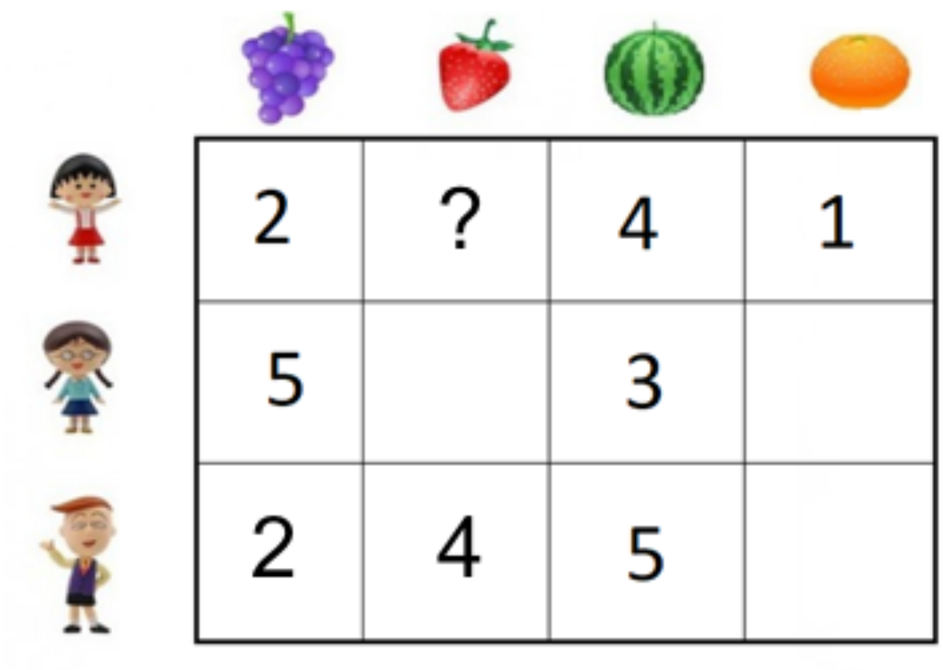
| | | | |
|---|---|---|---|
| 2 | ? | 4 | 1 |
| 5 | | 3 | |
| 2 | 4 | 5 | |

# **Content-based** recommendation

# Content-based

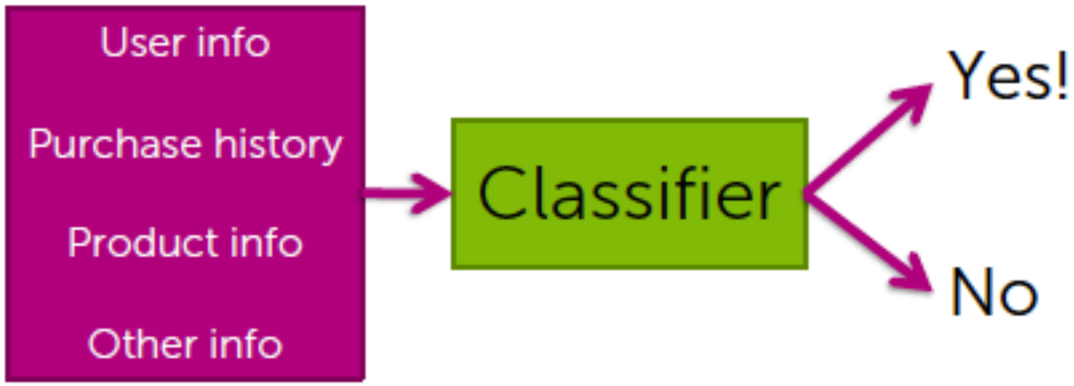
| FEATURE | VALUE |
|---|---|
| is_round | 1 |
| is_juicy | 1 |
| average_price | $1.99/lb |

Key aspect:
Have common features for each item



# Content-Based Recommendation

- Reduce per-user prediction to supervised prediction problem




Challenge: What features are necessary?

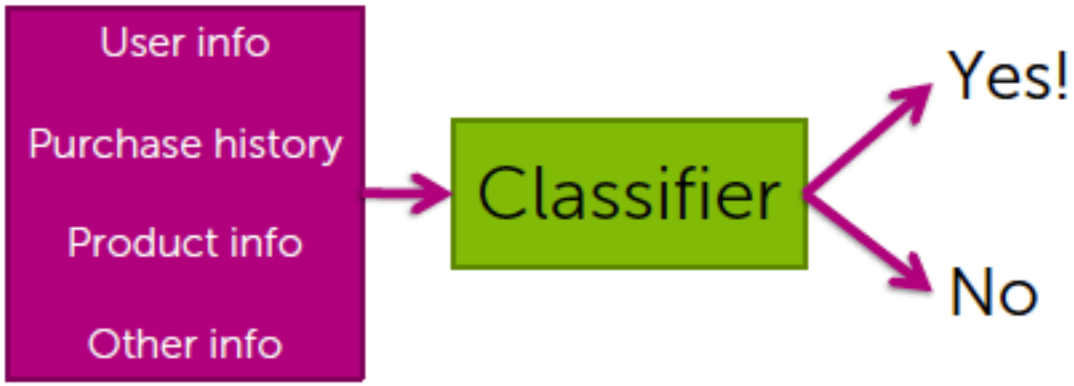
Fig. Credit: Emily Fox (UW)

# Possible Per-Item Features

If the item is a …

- **Movie**
  - *Possible features*: Set of actors, director, genre, year
- **Document**
  - *Possible features:* Bag of words, author, genre, citations
- **Product**
  - *Possible features:* Company name, description text

# Content-Based Recommender




**Pros:**

- Personalized:
  Considers user info & purchase history
- Features can capture context:
  Time of the day, what I just saw,...
- Even handles limited user history:
  Age of user, ...

**Cons:**

- Features may not be available
- Often doesn't perform as well as collaborative filtering methods (next)

Fig. Credit: Emily Fox (UW)

# Collaborative filtering

Two types:
- Neighbor methods
- Latent factor methods

# Neighbor Methods for Collaborative Filtering

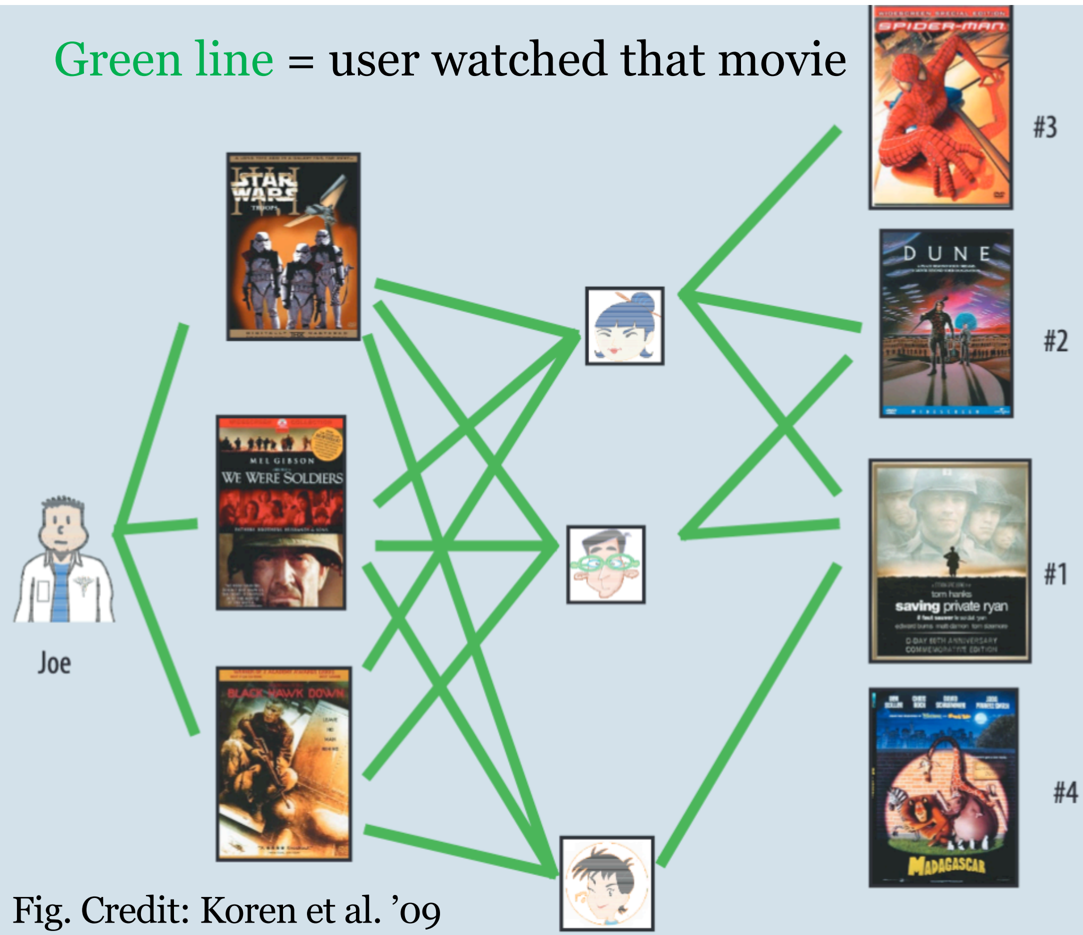



Fig. Credit: Koren et al. '09

Nearest Neighbor Recommendation

To find new movies to recommend to Joe

1) Find neighbors with similar preferences (other users who also liked movies that Joe likes)

2) Recommend movies that these neighbors liked

# Latent Factor Methods for Collaborative Filtering

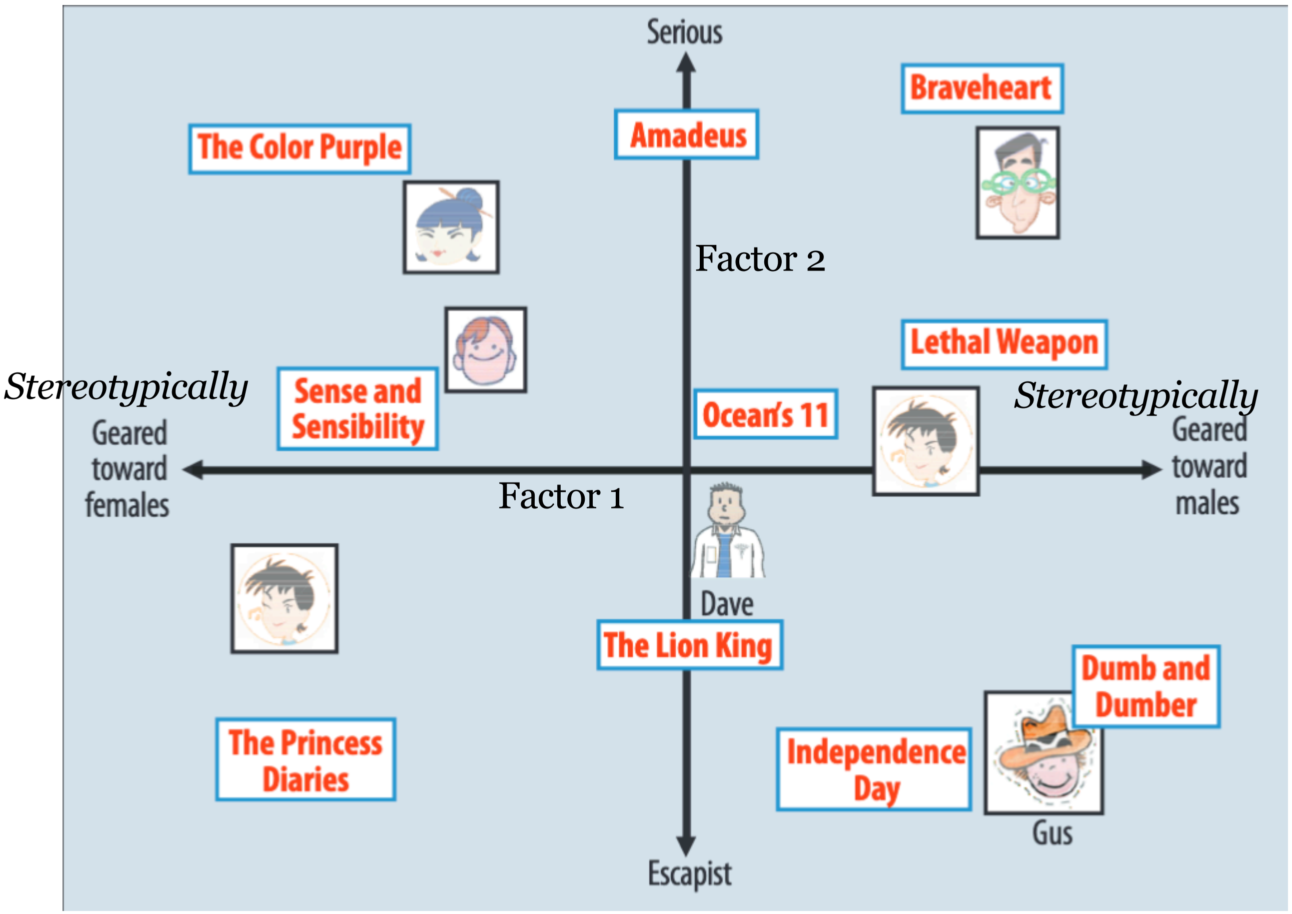



Fig. Credit: Koren et al. '09

Assumption:
- Both movies and users can be explained via a low-dimensional space

Latent Factor Recommendation

To find new movies to recommend to Joe

1) Find Joe's embedding vector in the learned "factor" space

2) Recommend movies with similar embedding vectors

# Latent Factor Model: **Prediction**

Assume a known number of factors K

- User *i* represented by vector $u_i \in \mathbb{R}^K$
- Item *j* represented by vector $v_j \in \mathbb{R}^K$

We predict the rating y for user-item pair (i,j) as:

$$\hat{y}_{ij} = \underbrace{\sum_{k=1}^{K} u_{ik} v_{jk}}_{u_i^T v_j}$$

Inner product of:

- User vector
- Item vector

Intuition:

Two items with similar v vectors get similar ratings from the same user;
Two users with similar u vectors give similar ratings to the same item

# Cartoon View of Matrix Factorization with 2 latent factors

*movies*

*users*

| | | NERO | JULIUS CAESAR | CLEOPATRA | SLEEPLESS IN SEATTLE | PRETTY WOMAN | CASABLANCA |
|---|---|---|---|---|---|---|---|
| HISTORY | 1 | 1 | 1 | 1 | 0 | 0 | 0 |
| | 2 | 1 | 1 | 1 | 0 | 0 | 0 |
| | 3 | 1 | 1 | 1 | 0 | 0 | 0 |
| BOTH | 4 | 1 | 1 | 1 | 1 | 1 | 1 |
| ROMANCE | 5 | -1 | -1 | -1 | 1 | 1 | 1 |
| | 6 | -1 | -1 | 1 | 1 | 1 | 1 |
| | 7 | -1 | -1 | -1 | 1 | 1 | 1 |

$Y$

$\approx$

| | HISTORY | ROMANCE |
|---|---|---|
| 1 | 1 | 0 |
| 2 | 1 | 0 |
| 3 | 1 | 0 |
| 4 | 1 | 1 |
| 5 | -1 | 1 |
| 6 | -1 | 1 |
| 7 | -1 | 1 |

U

| NERO | JULIUS CAESAR | CLEOPATRA | SLEEPLESS IN SEATTLE | PRETTY WOMAN | CASABLANCA | |
|---|---|---|---|---|---|---|
| 1 | 1 | 1 | 0 | 0 | 0 | HISTORY |
| 0 | 0 | 1 | 1 | 1 | 1 | ROMANCE |

$V^T$

$$Y \approx UV^T$$

Fig. Credit: Aggarwal 2016
By way of M. Gormley

# Supervised Learning vs Unsupervised Matrix Completion

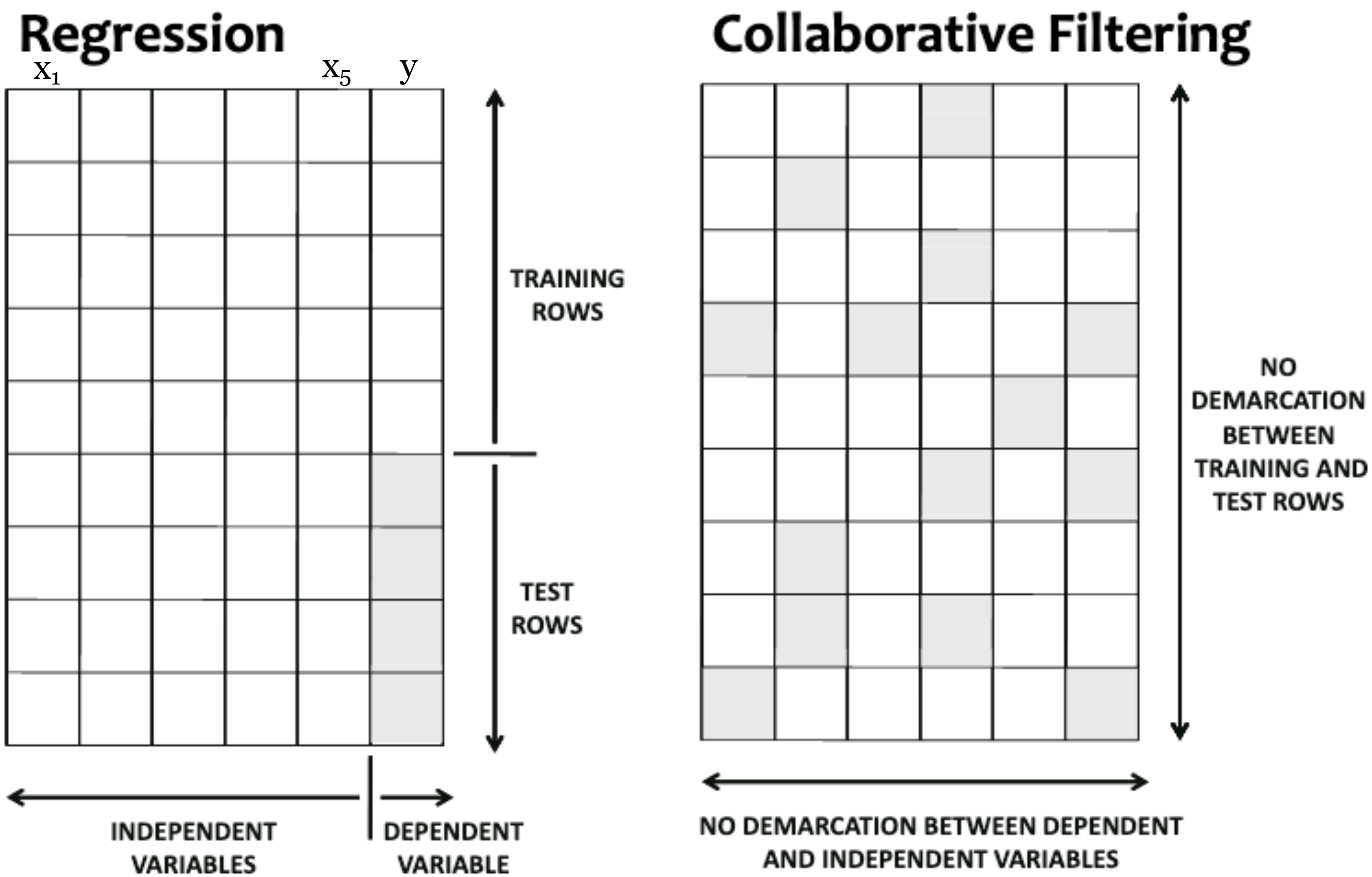
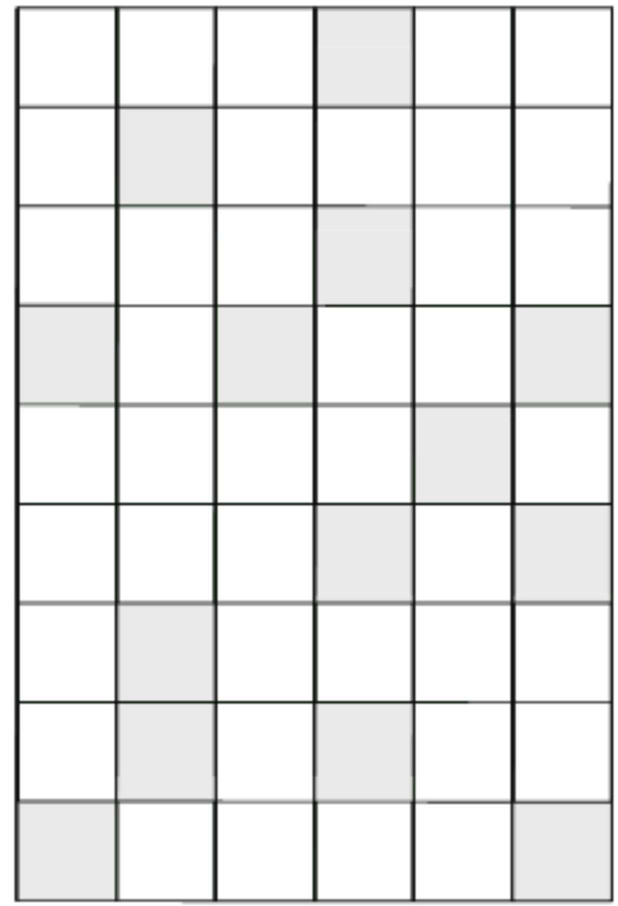



Fig. Credit: Aggarwal 2016
By way of M. Gormley

# Setting up Collaborative Filtering task

Real data will have known and unknown entries

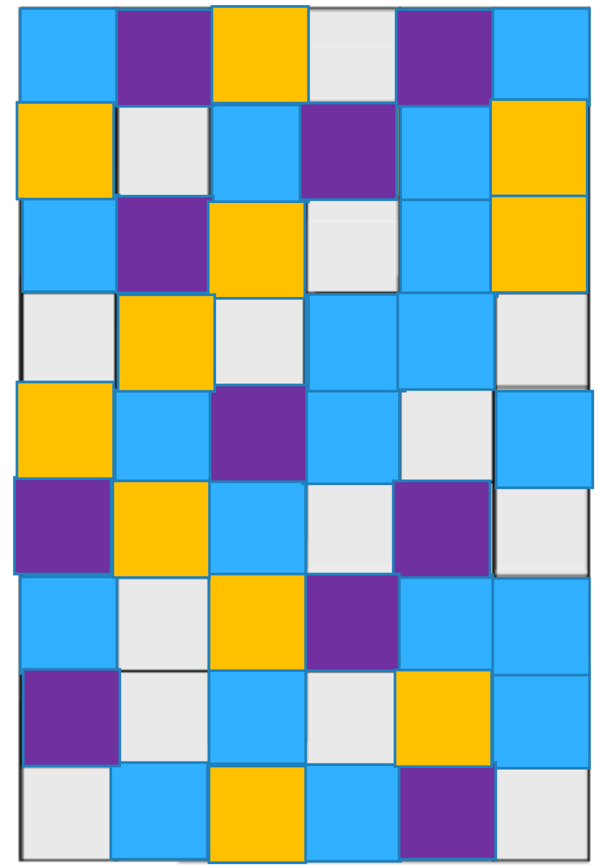


# Setting up Collaborative Filtering task

Real data will have known and unknown entries

Divide known user-item pairs at random into:

- Training
- Validation
- Test



Assumption (for Project C): We only care about predictions among known sets of users and items.
Do not worry about any new users or new items. (Obviously, in real world need to handle new users/items)

# Latent Factor Model: **Training**

- Find parameters that minimize squared error

$$\min_{u_i \in \mathbb{R}^K, v_j \in \mathbb{R}^K} \sum_{i,j \in \mathcal{I}^{\mathrm{train}}} (y_{ij} - u_i^T v_j)^2$$

Which pairs do we use?

Only the blue squares in the matrix Y

Squared error between
- True rating
- Predicted rating

- How to optimize?
  - **Stochastic gradient descent**
  - Use random minibatch of user-item pairs

# Improvement 1: Include intercept parameters!

- Overall "average rating" $\mu$
- Per-user scalar $b_i$
- Per-item scalar $c_j$

$$\hat{y}_{ij} = \mu + b_i + c_j + \sum_{k=1}^{K} u_{ik} v_{jk}$$

Why include these? **Improve accuracy**

Some items just more popular

Some users just more positive

# Improvement 2: Regularize latent factors

$$\min_{\mu,b,c,\{u_i\}_{i=1}^{N},\{v_j\}_{j=1}^{M}} \quad \alpha\Big(\sum_j \sum_k v_{jk}^2 + \sum_i \sum_k u_{ik}^2\Big) + \sum_{i,j \in \mathcal{I}^{\text{train}}} (y_{ij} - \mu - b_i - c_j - u_i^T v_j)^2$$

***Sum of squares penalty on u and v vectors***

Select penalty strength alpha on validation set

## Why do this? **Avoid overfitting**

Recall that:

U has N * K parameters and V has M * K parameters

Could overfit if training size is small even for modest K values

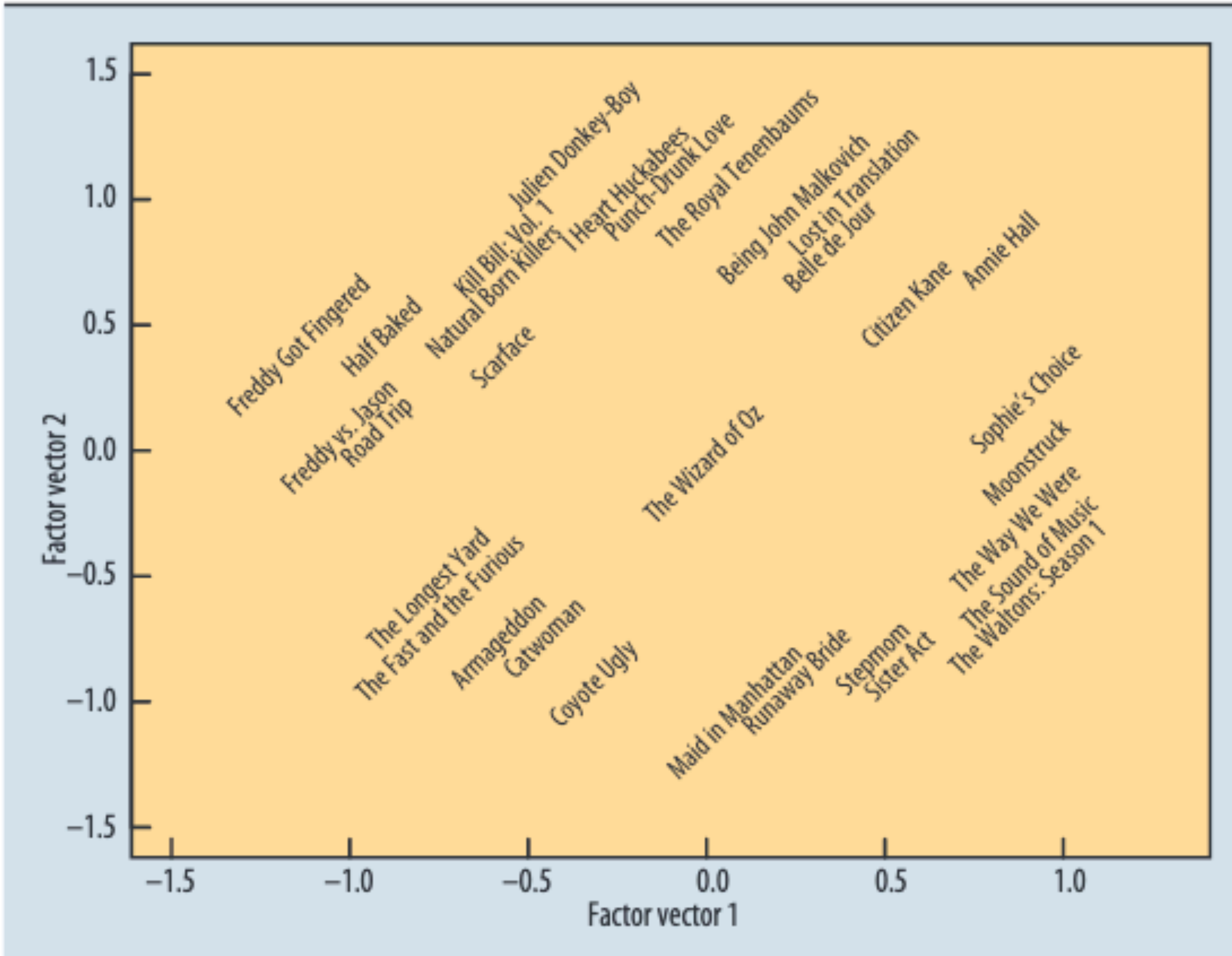



**Figure 3. The first two vectors from a matrix decomposition of the Netflix Prize data. Selected movies are placed at the appropriate spot based on their factor vectors in two dimensions. The plot reveals distinct genres, including clusters of movies with strong female leads, fraternity humor, and quirky independent films.**

Fig. Credit: Koren et al. ’09

factorization. Movies are placed according to their factor vectors. Someone familiar with the movies shown can see clear meaning in the latent factors. The first factor vector ($x$-axis) has on one side lowbrow comedies and horror movies, aimed at a male or adolescent audience (*Half Baked*, *Freddy vs. Jason*), while the other side contains drama or comedy with serious undertones and strong female leads (*Sophie's Choice*, *Moonstruck*). The second factorization axis ($y$-axis) has independent, critically acclaimed, quirky films (*Punch-Drunk Love*, *I Heart Huckabees*) on the top, and on the bottom, mainstream formulaic films (*Armageddon*, *Runaway Bride*). There are interesting intersections between these boundaries: On the top left corner, where indie meets lowbrow, are *Kill Bill* and *Natural Born Killers*, both arty movies that play off violent themes. On the bottom right, where the serious female-driven movies meet the mainstream crowd-pleasers, is *The Sound of Music*. And smack in the middle, appealing to all types, is *The Wizard of Oz*.


Comment on previous slide
Credit: Koren et al. '09

# Latent Factor Model Performance

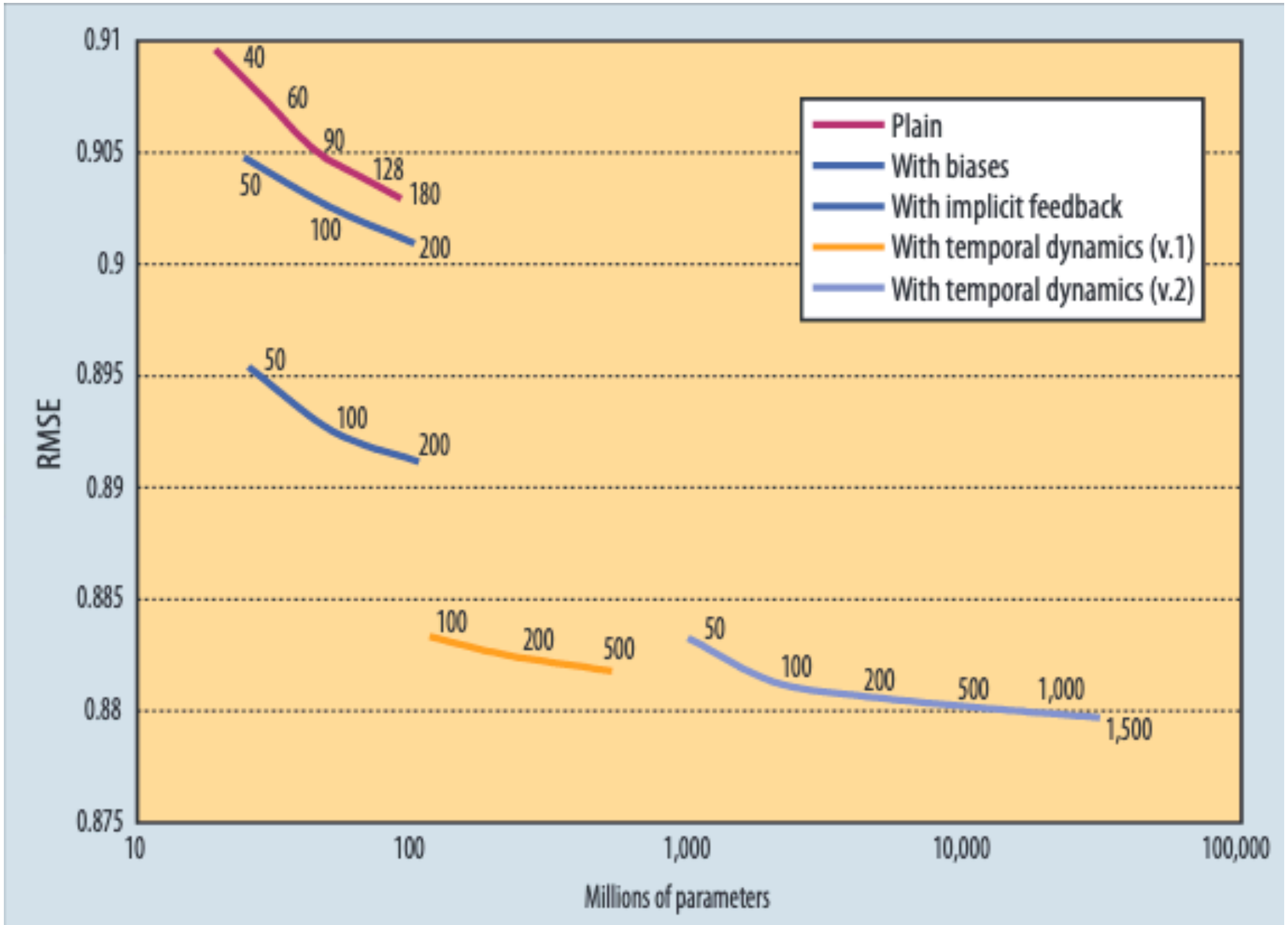



Fig. Credit: Koren et al. '09

# Limitations: Cold Start Issue

- New user entering the system
  - Hard for both content-based and matrix factors
  - Matching similar users
  - Trial-and-error
- New item entering the system
  - Easy with per-user content-based recommendation
    - IF easy to get the item's feature vector
  - Hard with matrix factorization
    - Trial-and-error

# Summary of Methods

# Task: Recommendation

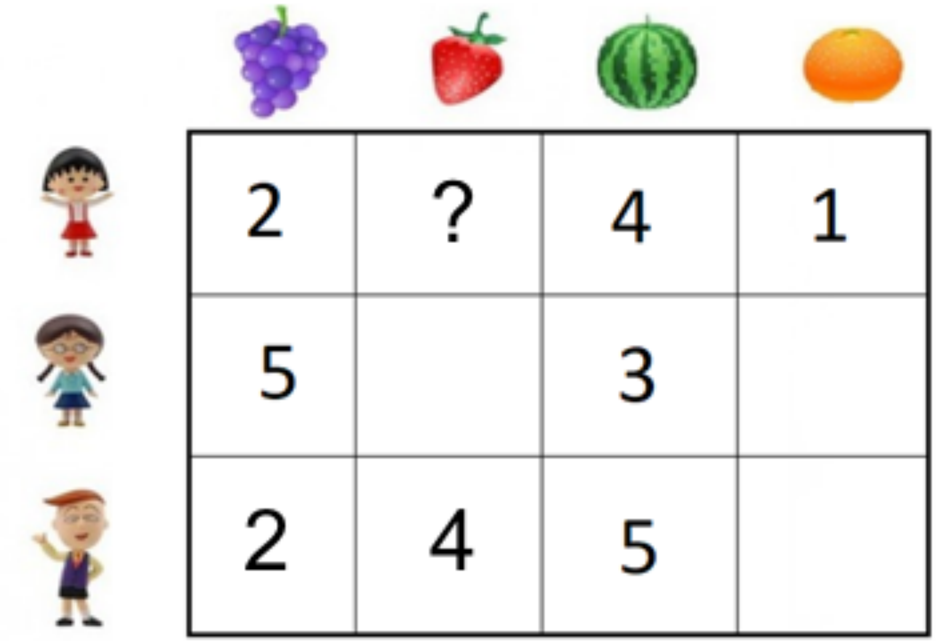
**Supervised Learning**

Content-based filtering

**Unsupervised Learning**

Collaborative filtering

Reinforcement Learning



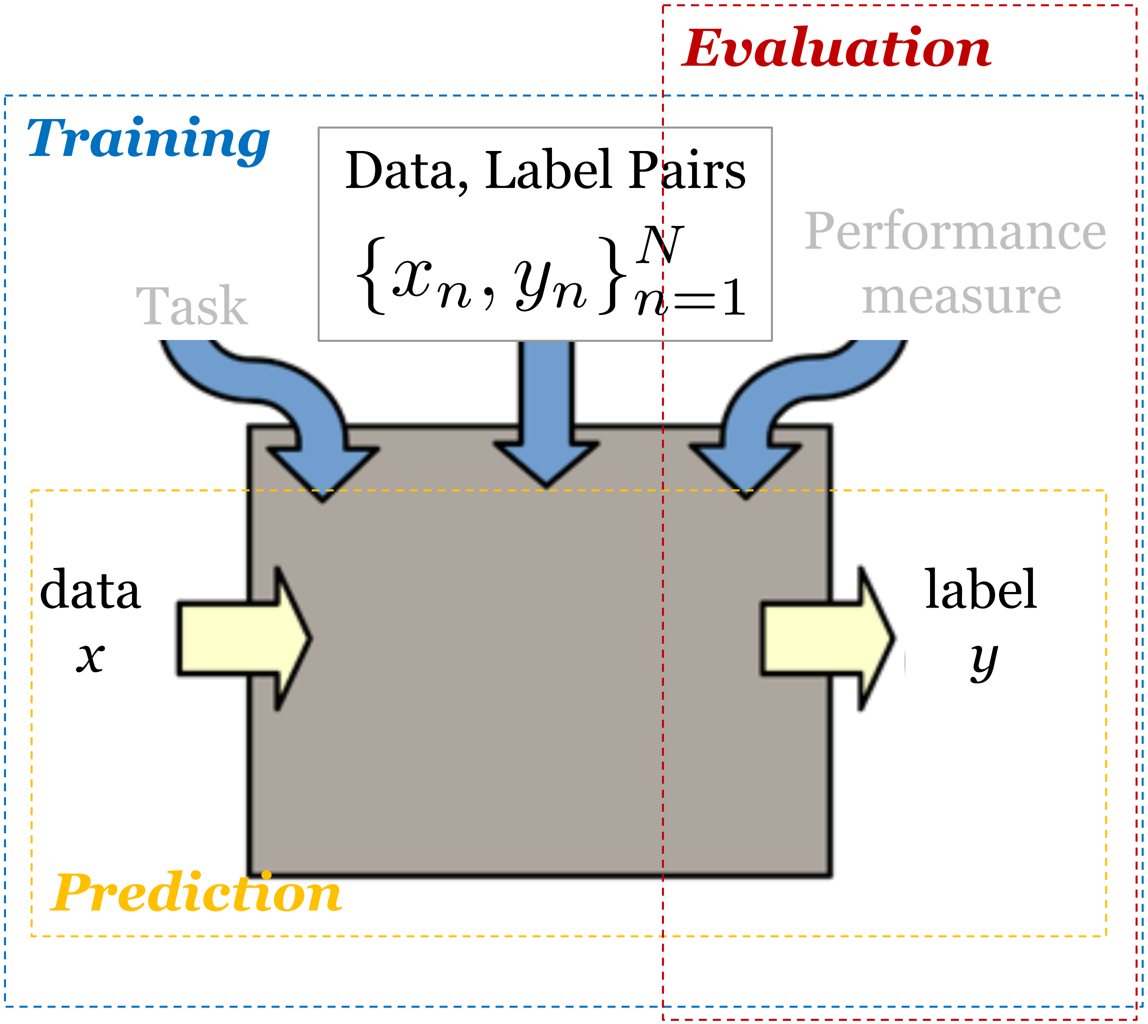
| | | | |
|---|---|---|---|
| 2 | ? | 4 | 1 |
| 5 | | 3 | |
| 2 | 4 | 5 | |

# Recall: Supervised Method

Supervised Learning

Unsupervised Learning

Reinforcement Learning



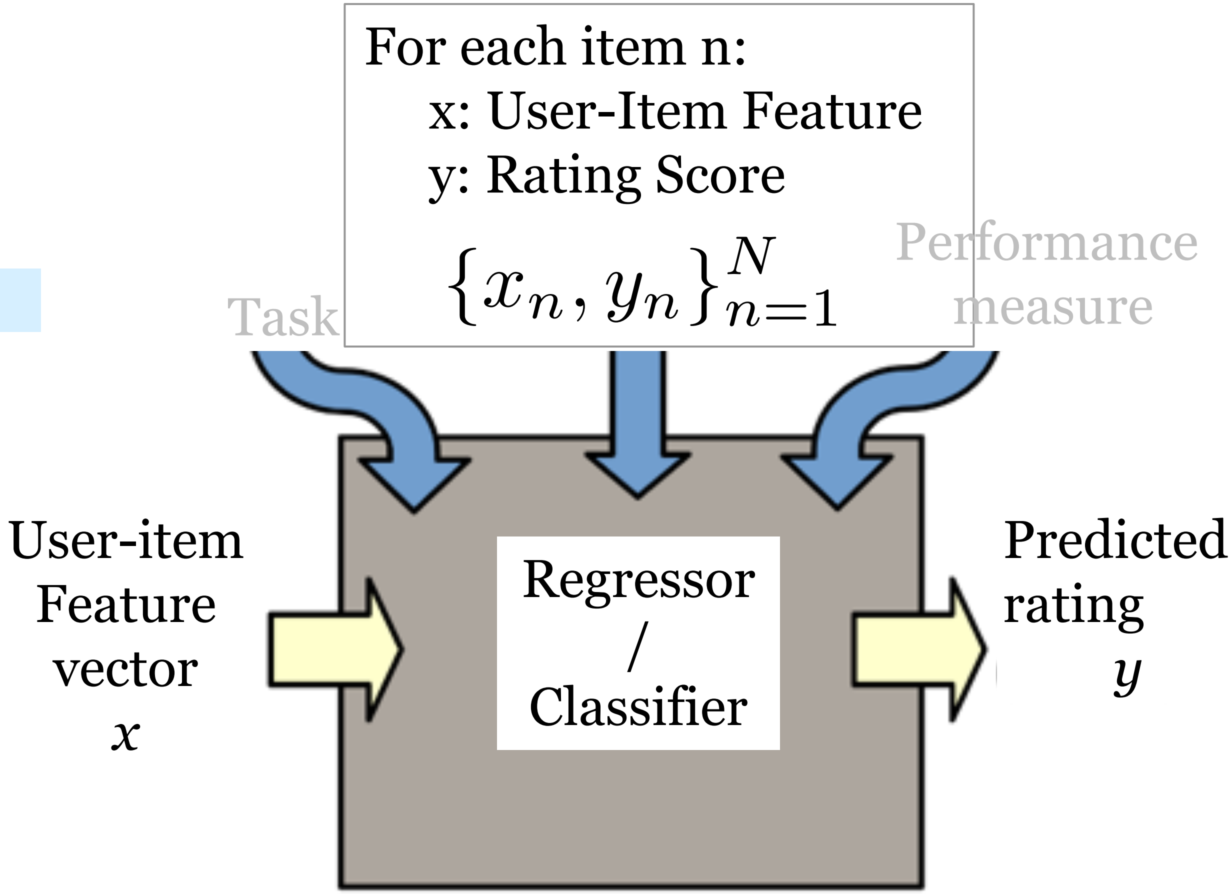
# Today: Per-User Predictor

Supervised Learning

**Content-based filtering**

Unsupervised Learning

Reinforcement Learning

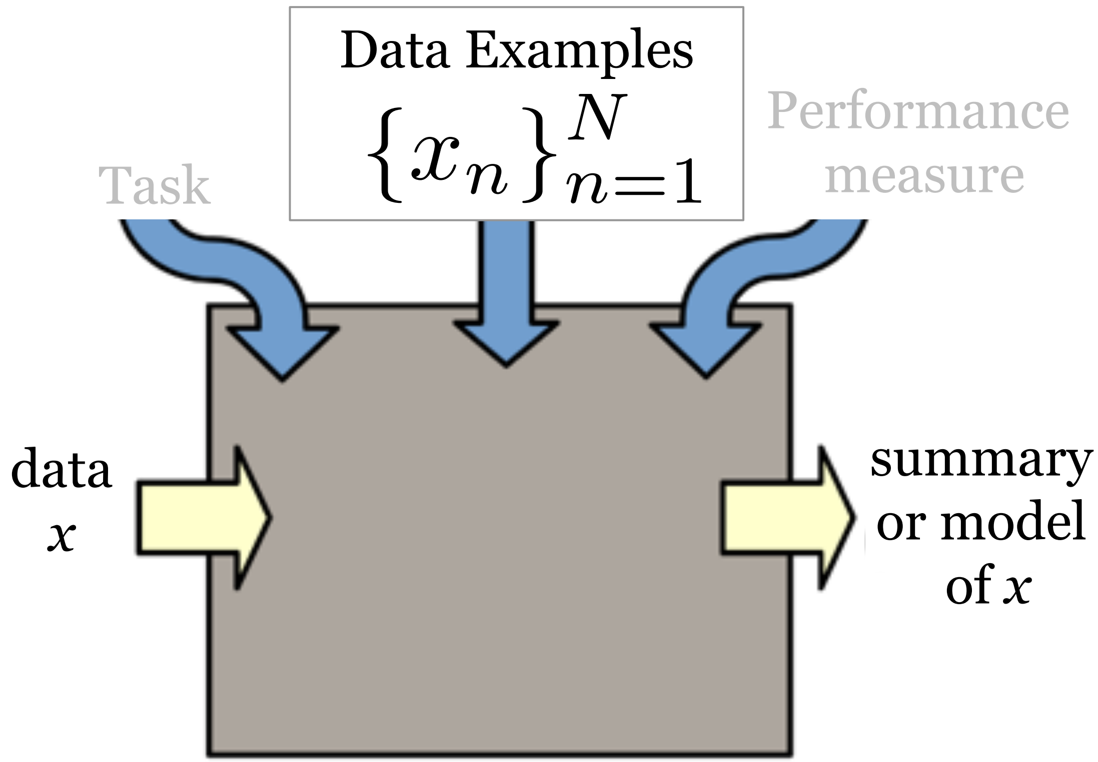


# Recall: Unsupervised Method

Supervised Learning

**Unsupervised Learning**

Reinforcement Learning



# Today: Matrix Factorization

Supervised Learning

**Unsupervised Learning**

**Collaborative filtering**

Reinforcement Learning