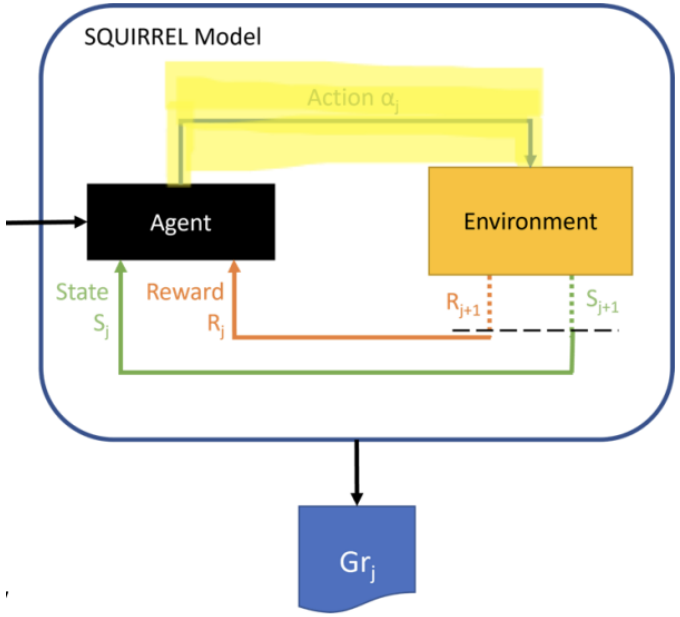
# SDAA Sequential Group Recommendations

**Satisfaction and Disagreement Aware Aggregation**

- Students: Oskari Perikangas, Xiaosi Huang
- Course: DATA.ML.360-2025-2026-1 Recommender Systems
- Date: November 10, 2025



# From Static to Dynamic Group Recommendations

The Limitations of User-Based Collaborative Filtering and Group Recommendations

**Example**: Movie 1276 predictions
- System predicts User 1 would rate it: 5.00
- System predicts User 2 would rate it: 3.76
- Disagreement : 1.24

**Challenges:**
- Average method and Least misery methods don't learn from previous recommendations
- Some users stay unhappy over multiple rounds
- No adaptation to group dynamics

**Solution:** SDAA framework
- Recommends movies over multiple rounds
- Adjusts strategy based on user satisfaction
- Balances group happiness and individual fairness

# SDAA Implementation

1st satisfaction scores: Decides which movies to recommend to the group

$$score(G, i, j) = (1 - \alpha_j) * avgG(G, i, j) + \alpha_j * leastG(G, i, j)$$

2nd Alpha Adaptation : Measures how unfair the previous round was

$$\alpha_j = max_{u \in G} sat(u, Gr_{j-1}) - min_{u \in G} sat(u, Gr_{j-1})$$

3rd Reward Evaluation: Balances group satisfaction and fairness

$$R_{sd}(RS^j) = 2 \frac{groupSatO(RS^j) * (1 - groupDisO(RS^j))}{groupSatO(RS^j) + (1 - groupDisO(RS^j))}$$

# How Alpha adapts strategy :

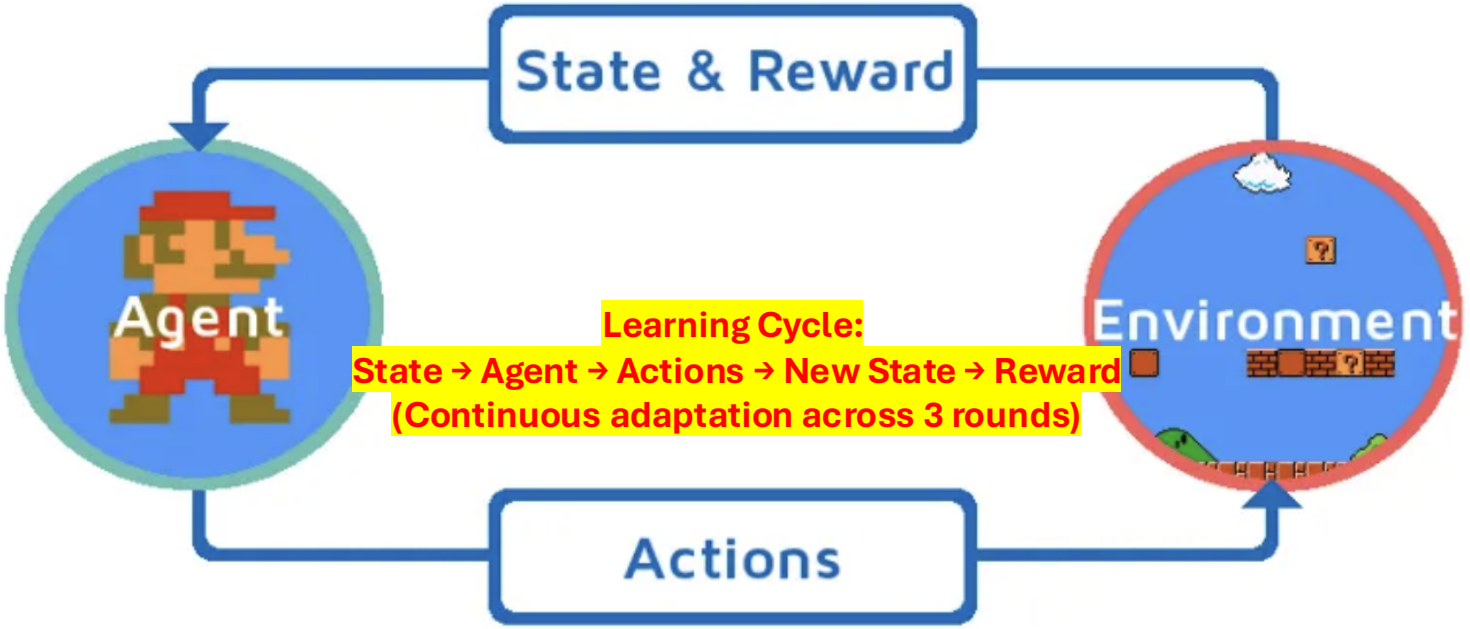
Small $\alpha_j$ (users equally satisfied) → More weight on Average method
Large $\alpha_j$ (satisfaction differences) → More weight on Least Misery

- Project result:
- Round 1: α = 0.2765 → 72% Average, 28% Least Misery
- Round 2: α = 0.4216 → 58% Average, 42% Least Misery
- Round 3: α = 0.4415 → 56% Average, 44% Least Misery

# SQUIRREL Reinforcement Learning Framework

State & Reward
Round 1: Satisfaction {1:0.9893, 414:0.7399, 599:0.5677}
Round 2: Satisfaction {1:0.9292, 414:0.6647, 599:0.4876}
Round 3: Satisfaction {1:0.9216, 414:0.7228, 599:0.5337}
Reward: 0.6590 → 0.6391 → 0.6477

SDAA Recommender System
- Tracks satisfaction history
- Calculates alpha parameters
- Adapts aggregation strategy




**Learning Cycle:**
**State → Agent → Actions → New State → Reward**
**(Continuous adaptation across 3 rounds)**

Environment
- MovieLens dataset
- Users [1, 414, 599]
- 882 popular movies

Actions
Round 1: α=0.2765 → [293, 318, 353, 16, 364]
Round 2: α=0.4216 → [180, 454, 17, 497, 508]
Round 3: α=0.4415 → [541, 529, 337, 252, 524]

# Reward changes reveal system intelligence:

**Round 1 → Round 2 → Round 3**

**0.6590 → 0.6391 ↓ → 0.6477 ↑**

The temporary drop shows smart balancing,
not system failure,
keeps all users engaged long-term,
not just maximizes a number.