



Hi, I'm Kiran Vodrahalli.

## Highest-Order Bits:

- **Final year Ph.D. Candidate @Columbia**
- **Advisors:** Alex Andoni and Daniel Hsu
- **Research:** Theory & Practice of ML
- **Website:** <https://kiranvodrahalli.github.io>
- **Job Search:** Industry Research or Postdocs



# Main Research Topics:

## Resource-Efficient Learning:

- Sample & Time Complexity
- Sparse Models
- **Low-Rank Models**
- Streaming Settings

## Controllable & Interpretable Agents:

- **Platform Design**
- Online & Reinforcement Learning
- Algorithmic Game Theory



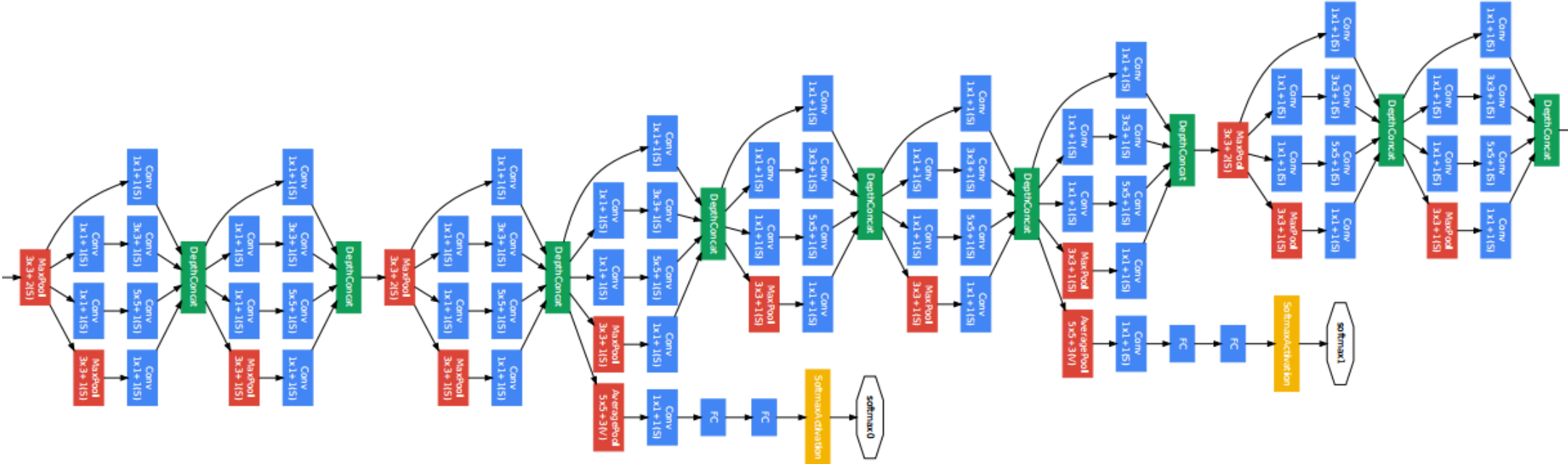


I also worked on applications:

- **Neuroscience**
- **NLP**
- **Robotics**
- **Economics**
- **Systems**

# Resource-Efficient Machine Learning

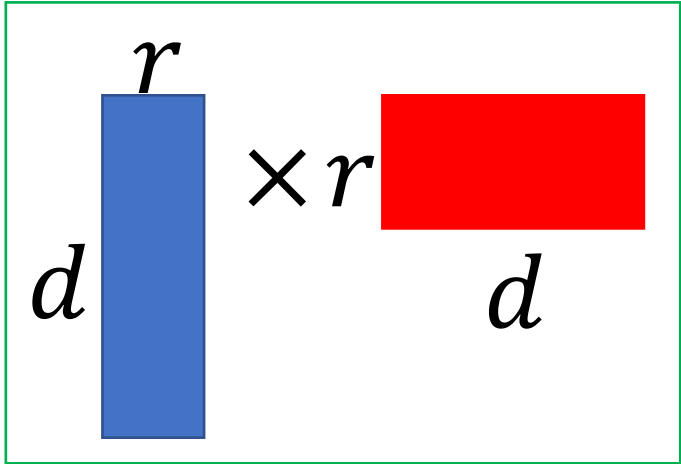
Modern ML challenge: **Very large models!**





# Sample & Time Complexity of Nonlinear Models

**Low-rank models?**

$$\sigma \left( \begin{array}{c} r \\ d \end{array} \times r \begin{array}{c} \\ d \end{array} x \right)$$
The diagram shows a mathematical expression for a low-rank model. It features a large sigma function symbol  $\sigma$  on the left, followed by a large opening parenthesis  $($ . Inside the parenthesis, there is a green-bordered box containing a matrix multiplication. The first matrix is a blue vertical rectangle with a height labeled  $r$  and a width labeled  $d$ . This is followed by a multiplication symbol  $\times$  and a red horizontal rectangle with a height labeled  $r$  and a width labeled  $d$ . To the right of the red rectangle is a large italicized  $x$ . The entire expression is closed with a large closing parenthesis  $)$ .

# Algorithms for Efficiently Learning Low-Rank Neural Networks

**Kiran Vodrahalli**, Rakesh Shivanna, Mahesh Sathiamoorthy, Sagar Jain, Ed Chi

Google Brain Research Internship

(in submission + arXiv soon!)



# Low Rank Deep Models

Replace full-rank layers with low-rank parameters.

Given weights of layer  $i$ :

$$W_i = U_i V_i^T$$

Standard initialization: **SVD** of full-rank init.

# Nonlinear Kernel Projection (NKP)

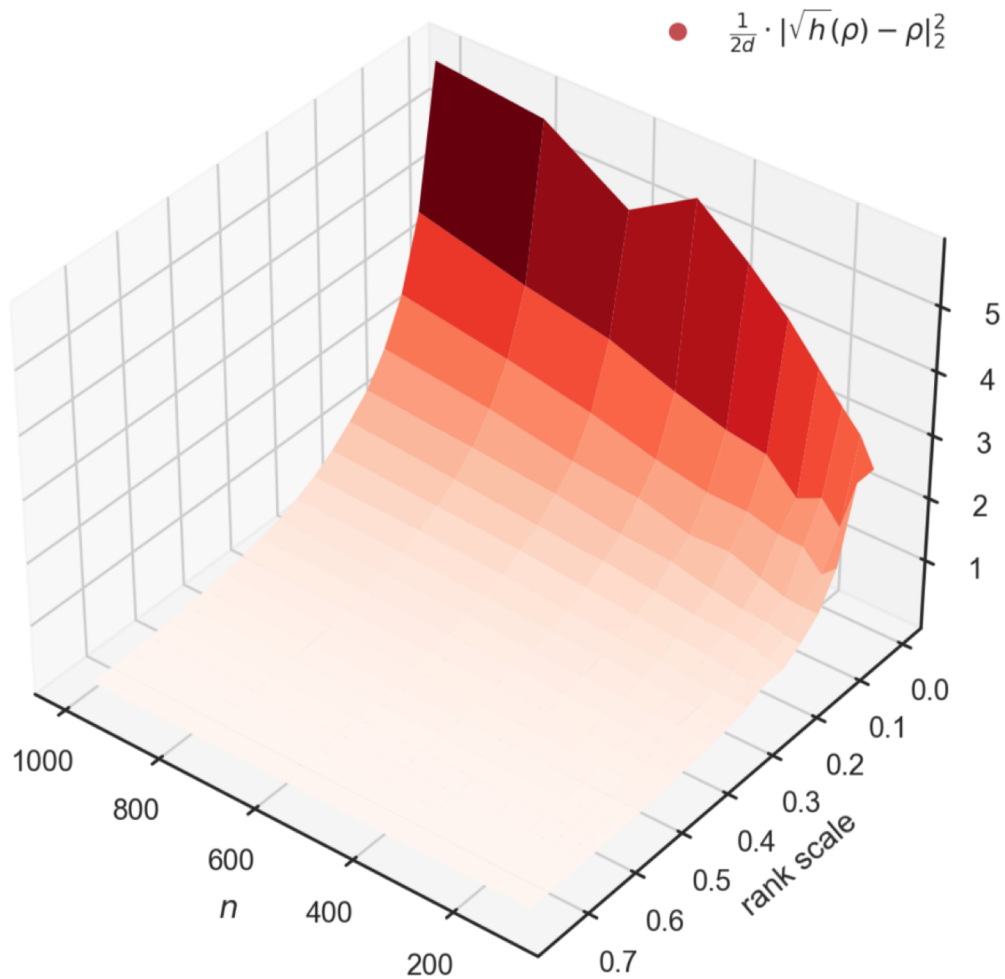
For each layer  $W \in R^{m \times n} \sim D$  with nonlinearity  $\sigma$ :

$$\min_{U \in R^{m \times r}, V \in R^{n \times r}} E_{x \sim N(0, I)} [\|\sigma(Wx) - \sigma(UV^T x)\|_2^2]$$

Empirical gains over SVD init!

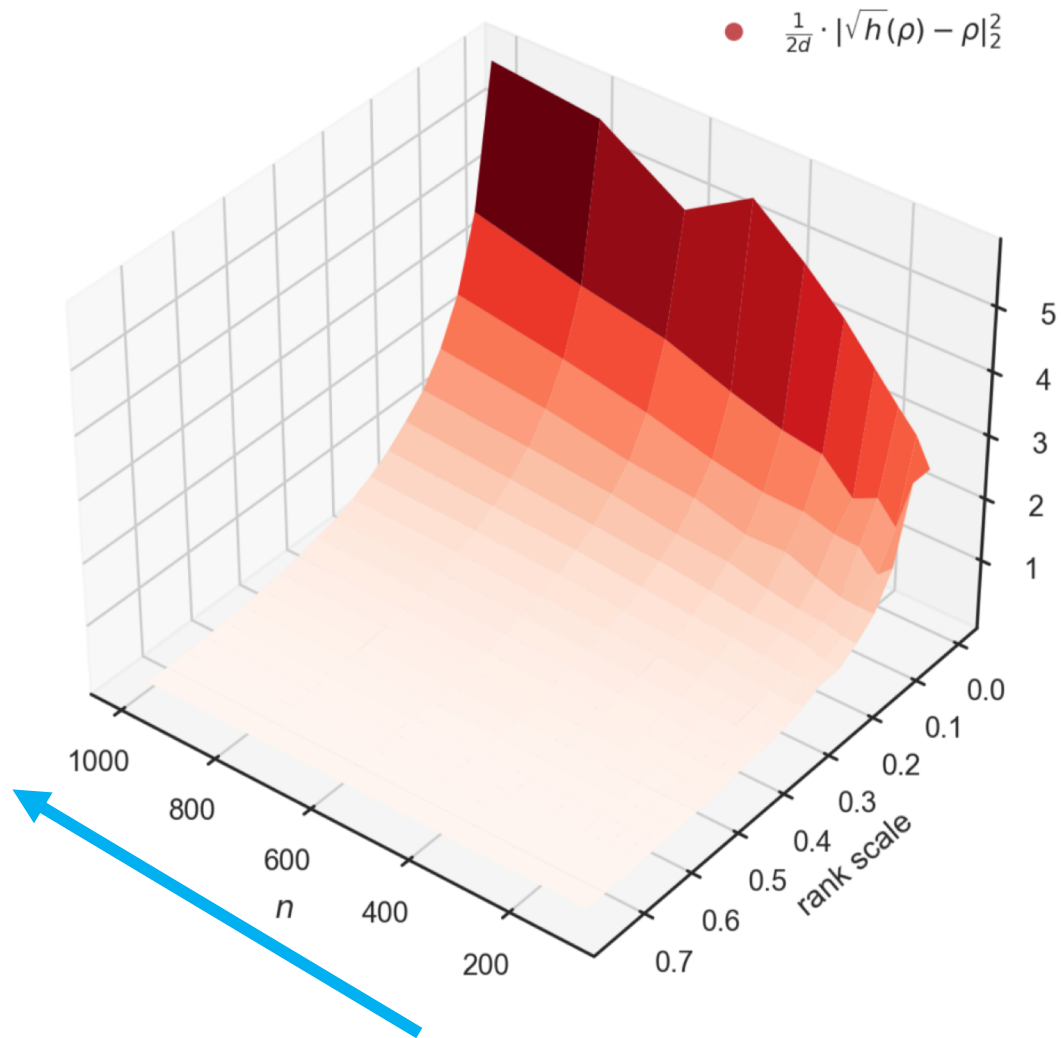


# Main Results



- Efficient optimal alg for NKP.
- Efficient *learning* alg for NKP.
- NKP outperforms SVD init.
- Especially with:
  - Larger width networks
  - Lower rank approx.

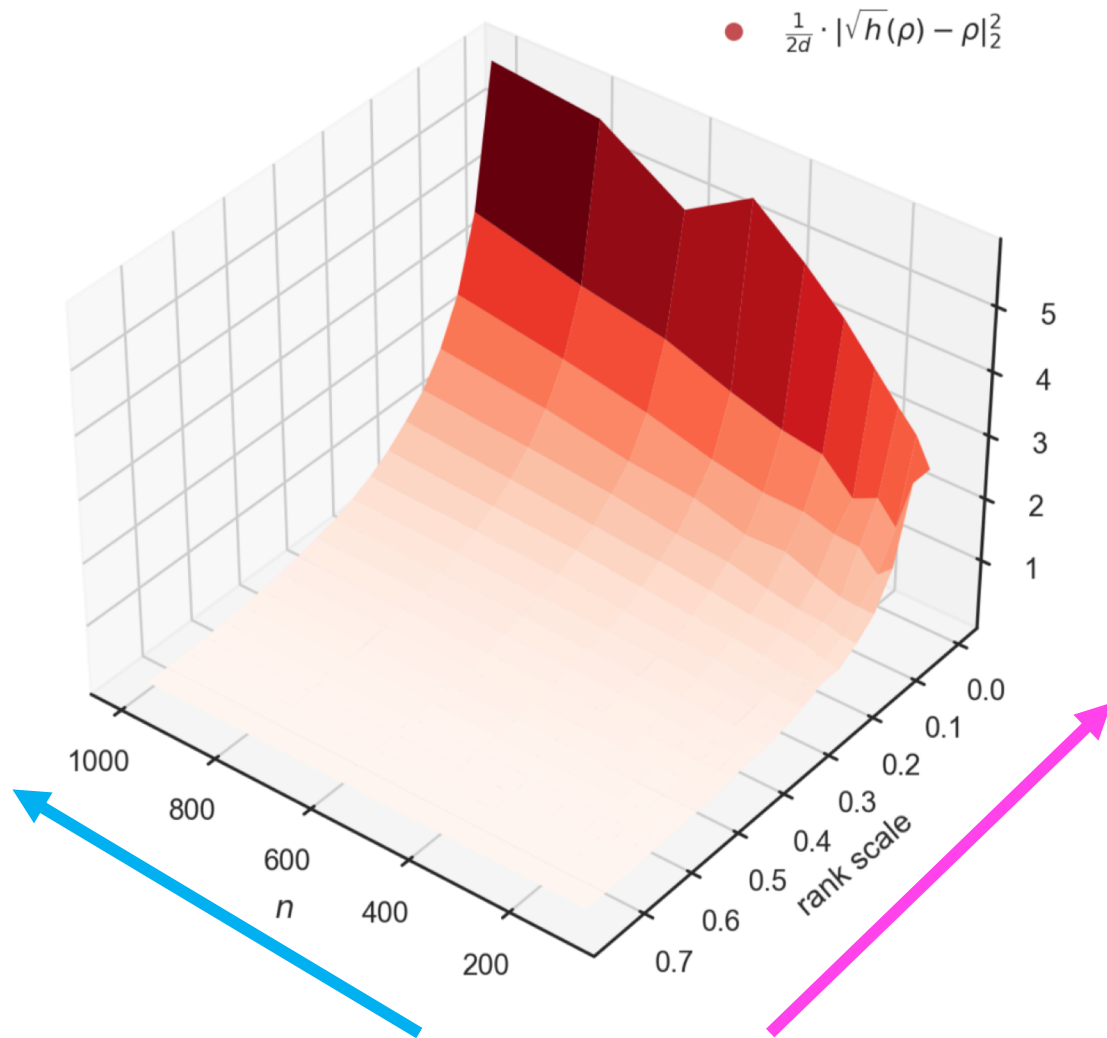
# Main Results



- Efficient optimal alg for NKP.
- Efficient *learning* alg for NKP.
- NKP outperforms SVD init.
- Especially with:
  - Larger width networks
  - Lower rank approx.



# Main Results



- Efficient optimal alg for NKP.
- Efficient *learning* alg for NKP.
- NKP outperforms SVD init.
- Especially with:
  - Larger width networks
  - Lower rank

# The Platform Design Problem

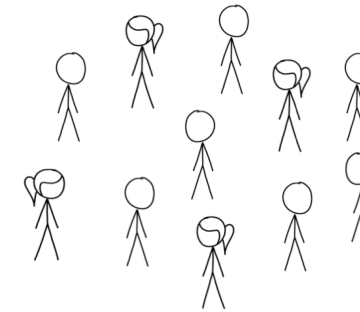
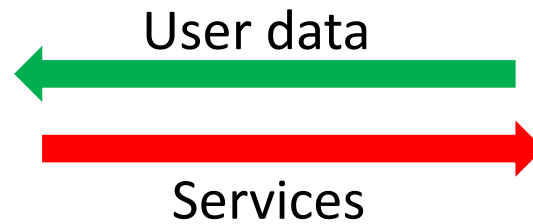
Christos Papadimitriou, **Kiran Vodrahalli**, Mihalis Yannakakis

WINE '21, NeurIPS Strategic ML Workshop '21

# Economics of the Online Firm



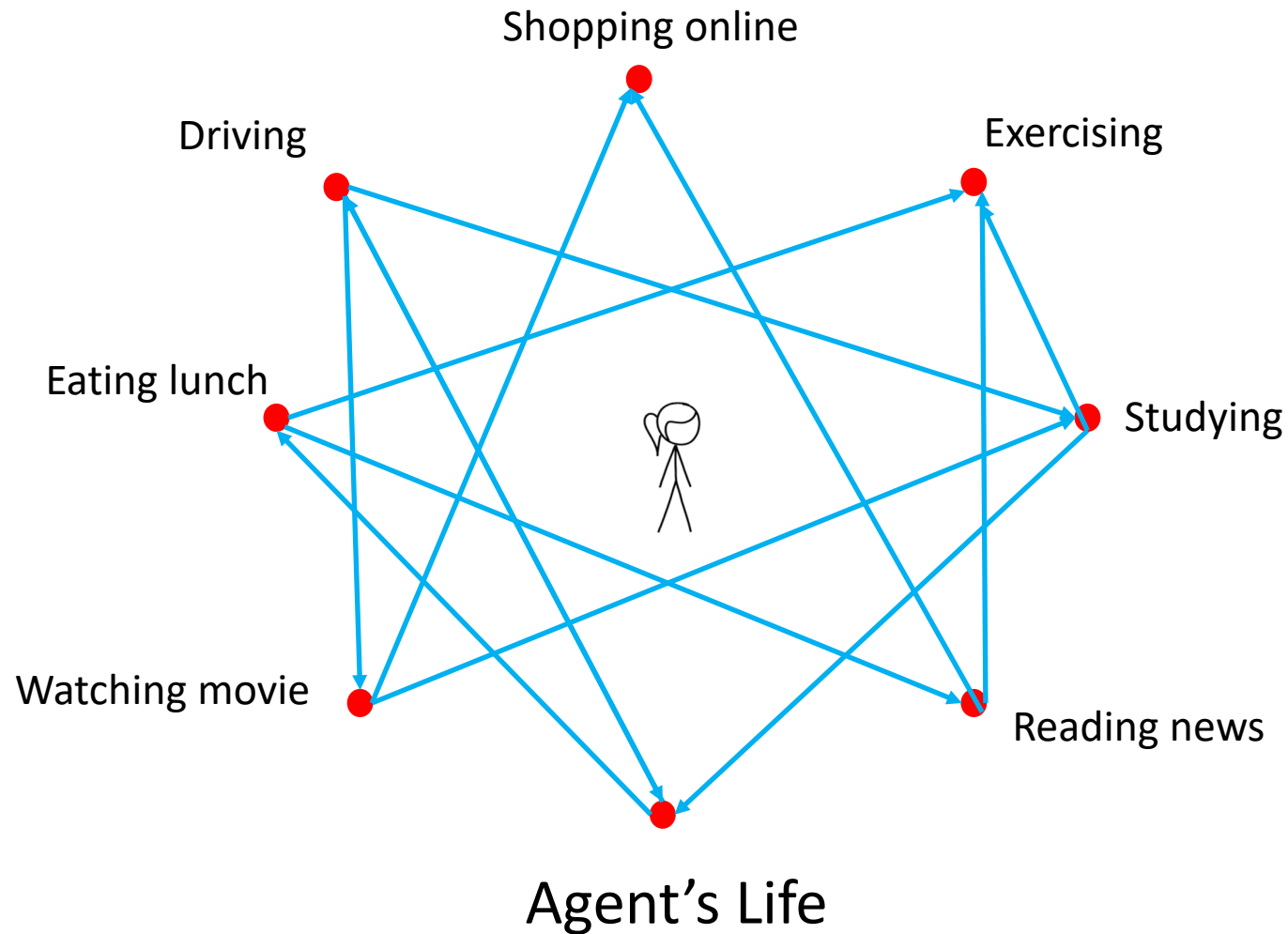
Online firm



Users

- User data feeds revenue
  - Better demand segmentation
  - Ad/recommendation revenue
  - Better models => better services
- Online services bring value
  - Convenience
  - Knowledge

# Picture of the General Case

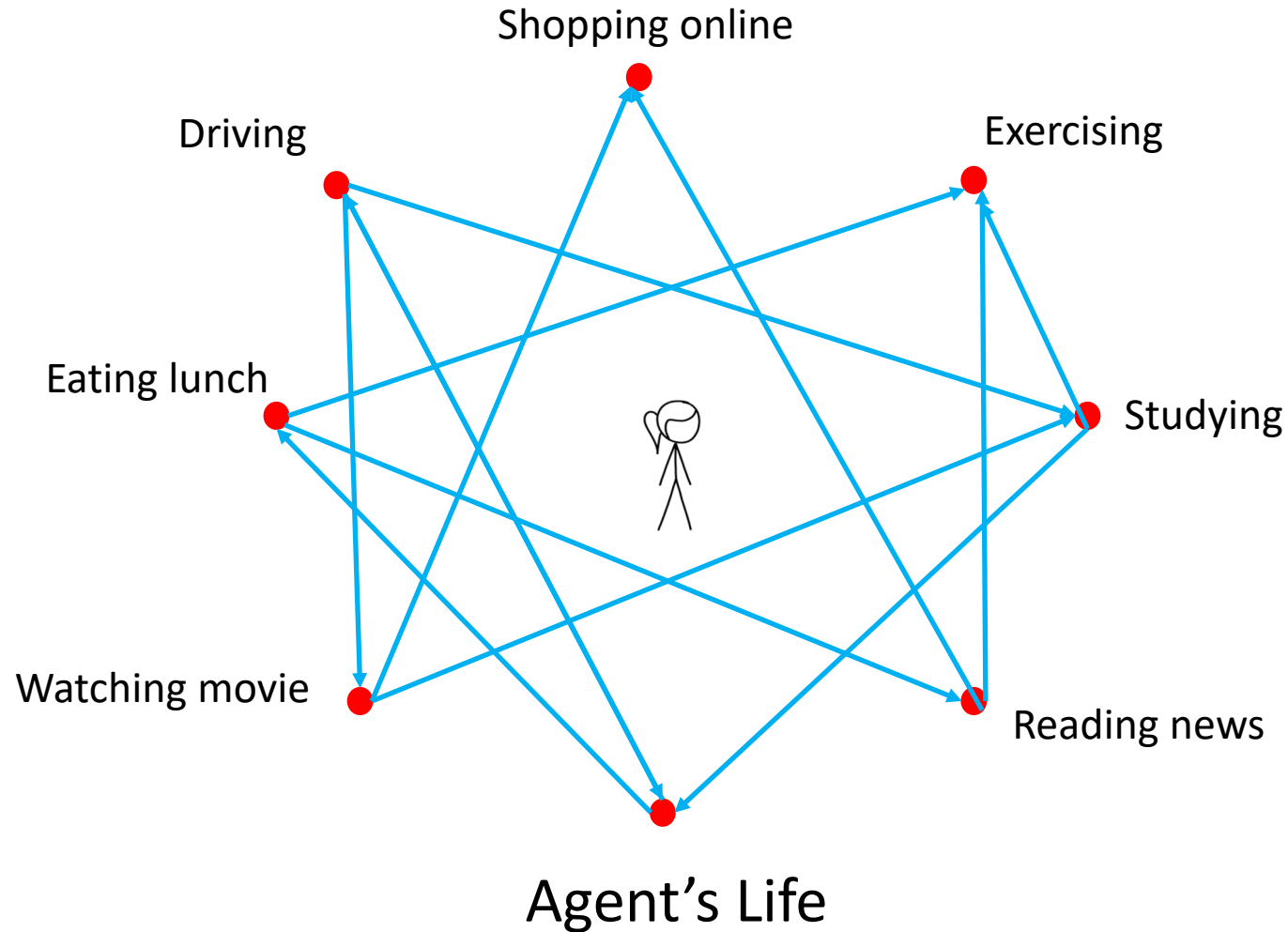


What platforms should I build?





# Picture of the General Case

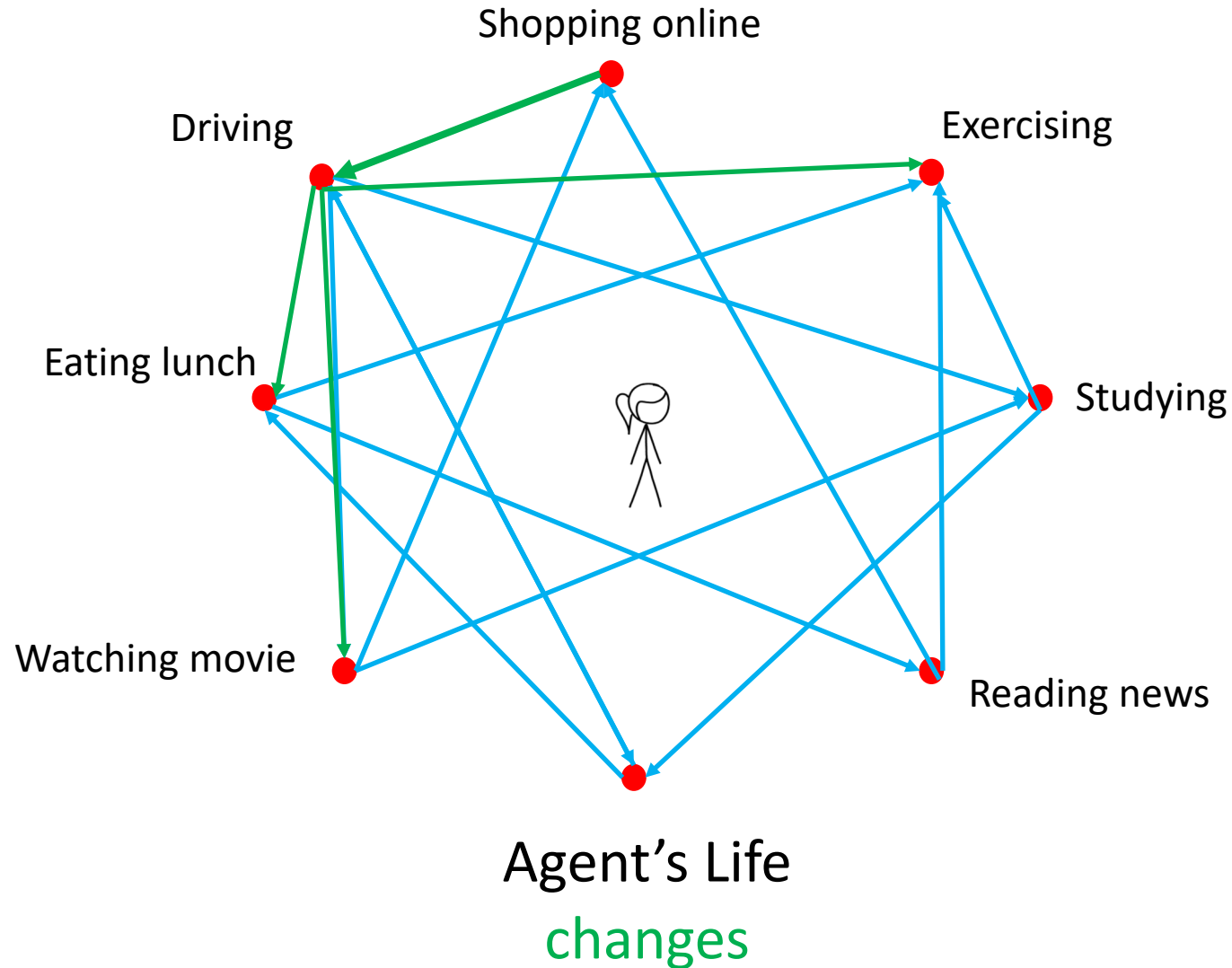


What platforms should I build?



At a cost, the firm can **add an opt-in action** to platforms they create (ex: Google Maps).

# Picture of the General Case



Maybe we should create Maps technology....



Builds platform Maps at a cost.



# A Stackelberg Game

- Designer moves first:
  - Adds **platforms** which modify transitions to an existing Markov Chain
- Agent moves second:
  - Receives **MDP** from Designer, plays optimal behavior
- Bi-level MDP optimization

# Grand Vision

- **Design environments** which generate useful, sampleable data
- **Model economics** of companies dependent on information economy
- **Model strategic behavior** of online firms and their users
- **Reinforcement learning** aided by environment design
- **Manipulation and resistance** of learning agents