# Reinforcement Learning

CS771: Introduction to Machine Learning

Purushottam Kar

# Announcements



**Doubt clearing session**: November 17 (Sunday), **6:00PM** KD101

**Practice end-sem paper** now available on website

**End-sem Exam**: November 18 (Monday), 4-7PM, L19, L20

*Syllabus is till whatever we cover today i.e. Nov 15 (Fri)*

*Bring your **institute ID card** with you – will lose time if you forget*

*Bring a **pencil, pen, eraser, sharpener** with you – we wont provide!*

*Answers to be written on question paper itself. If you write with pen and make a mistake, no extra paper. Final answer **must be in pen***

***Auditors cannot appear** for the mid-sem exam*

**Seating for end-sem exam**: assigned seating (announced on Piazza)

*Everyone has been assigned a lecture hall, and a seat number in that hall*

*All must go to their assigned hall – no extra space if you go to wrong hall*

# Reinforcement Learning



We have an environment $\mathcal{E}$ with which our ML/RL algo can interact

If we were to take a recsys example, a user may be our "environment"

*At any point of time* $t$*, env is in a certain* state $s_t \in \mathcal{S}$

The user has certain likes and certain dislikes

*Algo can perform an action* $a_t \in \mathcal{A}$ *from a set of allowed* actions

Recommend one of the products currently available for sale on the website

*In response,* $\mathcal{E}$ *changes its state to* $s_{t+1}$ *and gives algo a reward* $r_t$

Reward is a click/purchase, state change may be user discovering a new liked item

Goal: keep playing this game and maximize total reward $\sum_{t \geq 1} r_t$

Simplified form

*Current state of environment completely visible to algo – not true in RecSys*

*Next state* $s_{t+1}$ *a deterministic function of* $s_t, a_t$ *but this function is unknown*

*Reward* $r_t$ *depends only on next state* $s_{t+1}$ *e.g. if Mario jumps on a Koopa*

# Markovian Decision Process (MDP)



A more general formulation of this is an MDP

*Given action $a_t$ when in state $s_t$, next state/reward is not entirely deterministic*

*Next state is $s$ with prob $\mathbb{P}_{a_t}(s_t, s)$ - this distribution is not known to algo*

*Reward may depend on current + next state + action i.e. $r_t = R_{a_t}(s_t, s_{t+1})$*

A still more general formulation is a partially observed MDP (POMDP)

*In a POMDP, even the current state of $\mathcal{E}$ visible only partially to the algo*

**Policy**: a rule that tells us which action to take on which state $\pi: \mathcal{S} \rightarrow \mathcal{A}$

*RL algorithms try various ways to learn the best policy that maximizes reward*

*Definitely want to take an action $a_t$ such that $r_t$ is big but also want to ensure that we land on a state after taking this action so that $r_{t+1}, r_{t+2}, \ldots$ also big*

*Would be naïve to take action that gives large $r_t$ but subsequent rewards $0$ ☹*

# Value Function



Given a policy $\pi: \mathcal{S} \rightarrow \mathcal{A}$, we assign a *value* to this policy

*Tells us how much reward we get if we always play actions dictated by policy* $\pi$

*Assume for simplicity that state transitions and rewards are deterministic*

***Notation****: action* $a$ *in state* $s$ *takes us to state* $T(s,a)$ *and gives reward* $R(s,a)$

Notation: let $s_0 \coloneqq s$ be first state, $s_1 = T\big(s_0, \pi(s_0)\big)$ be state after playing according to $\pi$ for one step, $s_2 = T\big(s_1, \pi(s_0)\big)$ be state after playing using $\pi$ for two steps and so on …

*Possible defn of value* $R\big(s_0, \pi(s_0)\big) + R\big(s_1, \pi(s_1)\big) + R\big(s_2, \pi(s_2)\big) + \cdots$

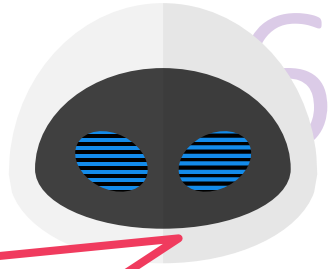
*Usually we care less about future rewards so* discount *them – let* $\gamma \in (0,1)$

$$V^{\pi}(s_0) \triangleq R\big(s_0, \pi(s_0)\big) + \gamma \cdot R\big(s_1, \pi(s_1)\big) + \gamma^2 \cdot R\big(s_2, \pi(s_2)\big) + \cdots$$

Note that we have $V^{\pi}(s_0) = R\big(s_0, \pi(s_0)\big) + \gamma \cdot V^{\pi}(s_1)$

*Called the* Bellman Equation. *For settings where state transitions and rewards are probabilistic, we replace the above using expectations*

As is obvious, RL algos try to learn policies that have high values for all states. For a given state $s$, $V^*(s)$ denotes the highest value obtainable when starting from that state (using any conceivable policy) i.e. $V^*(s) = \max_{\pi} V^{\pi}(s)$. Also let $\pi^*(s) = \arg\max_{\pi} V^{\pi}(s)$ denote the optimal policy which gets this value. Note that $V^*$ satisfies a recurrence $V^*(s) = \max_{a \in \mathcal{A}} \{R(s,a) + \gamma \cdot V^*(T(s,a))\}$



*by policy $\pi$*

*Assume for simplicity that state transitions and rewards are deterministic*

***Notation**: action $a$ in state $s$ takes us to state $T(s,a)$ and gives reward $R(s,a)$*

Not … $\pi$

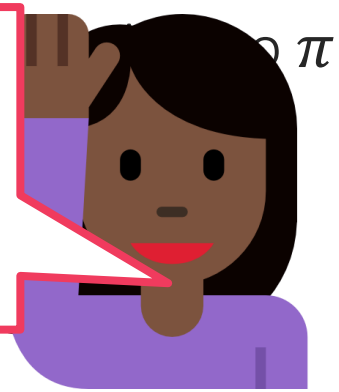
for …

*Possi…*

*Usua…*

$V^{\pi}(s_0) \triangleq R(s_0, \pi(s…$

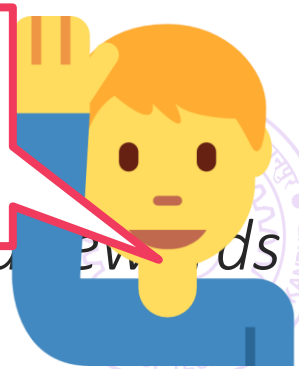
If someone could give us this magical optimal value function $V^*: \mathcal{S} \to \mathbb{R}$, then some math shows that we can come up with the best policy $\pi^*$ by simply defining $\pi^*(s) = \arg\max_{a \in \mathcal{A}} V^*(T(s,a))$. For probabilistic rewards and transitions, we use expectations in above definitions.



Note that we have $V$…

*Called the* Bellman Equation. *For settings where state transitions and rewards are probabilistic, we replace the above using expectations*

We can also show that if someone gives us the optimal policy $\pi^*$ then we could easily obtain the optimal value function $V^*$ by using the Bellman equations



# Value and Policy Iteration Algos



**Method 1**: **Value Iteration** (used if #states, #actions is small)

*Get a good estimate* $\hat{V}(s)$ *of* $V^*(s)$ *for all* $s \in \mathcal{S}$ *using pure exploration*

*Use this to define a policy* $\hat{\pi}(s) = \arg\max_{a \in \mathcal{A}} \{\hat{V}(T(s,a))\}$

*Not very nice just as pure exploration was not nice for bandits*

*Not feasible if number of states is large or action sets are large*

**Method 2**: **Policy Iteration** (alternating optimization)

*Start with a random policy* $\pi$*, find its value function* $V^{\pi}(\cdot)$ *using Bellman eqns*

*Update the policy as* $\pi(s) \leftarrow \arg\max_{a \in \mathcal{A}} \{V^{\pi}(T(s,a))\}$

*Repeat till converged or else till budget allows*

*Also not feasible for large state/action spaces and wasteful*

These are unsafe algos – will sacrifice many Marios while learning ☹

*Becomes unacceptable if we replace Mario with a real robot/drone*

# Q Learning



Reduces RL to regression/multiclass problems

Define a $Q$ function as follows

$$Q(s,a) = R(s,a) + \gamma \cdot V^*\big(T(s,a)\big)$$

*Recall that this means that* $V^*(s) = \max_{a \in \mathcal{A}} Q(s,a)$

*Note that this also means that the optimal action is* $\arg\max_{a \in \mathcal{A}} Q(s,a)$

Use a powerful ML algo (kernels, deepnets) to do one of the following

*Given (vector representations of) a state $s$ and action $a$, predict $Q(s,a)$*

Works when too many (infinite) states as well as actions

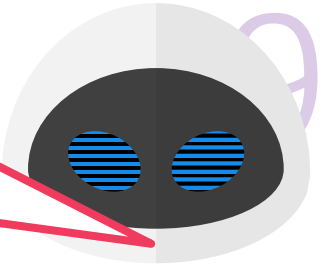
*Given (vector representation of) a state $s$, solve the multiclass problem of predicting the best action i.e.* $\arg\max_{a \in \mathcal{A}} Q(s,a)$

Works when too many (infinite) states but only a few actions

Note that this approach avoids value function/rewards altogether ☺

# Q Learning

Deep Q learning is a leading technique in reinforcement learning for several application domains – very convenient



Reduces RL to regression/multiclass problems

Define a $Q$ function as follows

$$Q(s,a) = R(s,a) + \gamma \cdot V^*\big(T(s,a)\big)$$

*Recall that this means that* $V^*(s) = \max_{a \in \mathcal{A}} Q(s,a)$

*Note that this also means that the optimal action is* $\arg\max_{a \in \mathcal{A}} Q(s,a)$

Use a powerful ML algo (kernels, deepnets) to do one of the following

*Given (vector representations of) a state $s$ and action $a$, predict $Q(s,a)$*

Works when too many (infinite) states as well as actions

*Given (vector representation of) a state $s$, solve the multiclass problem of predicting the best action i.e.* $\arg\max_{a \in \mathcal{A}} Q(s,a)$

Works when too many (infinite) states but only a few actions

Note that this approach avoids value function/rewards altogether ☺

# Episodic RL



In certain settings, instantaneous rewards i.e. rewards being given on every single action taken by algo is infeasible

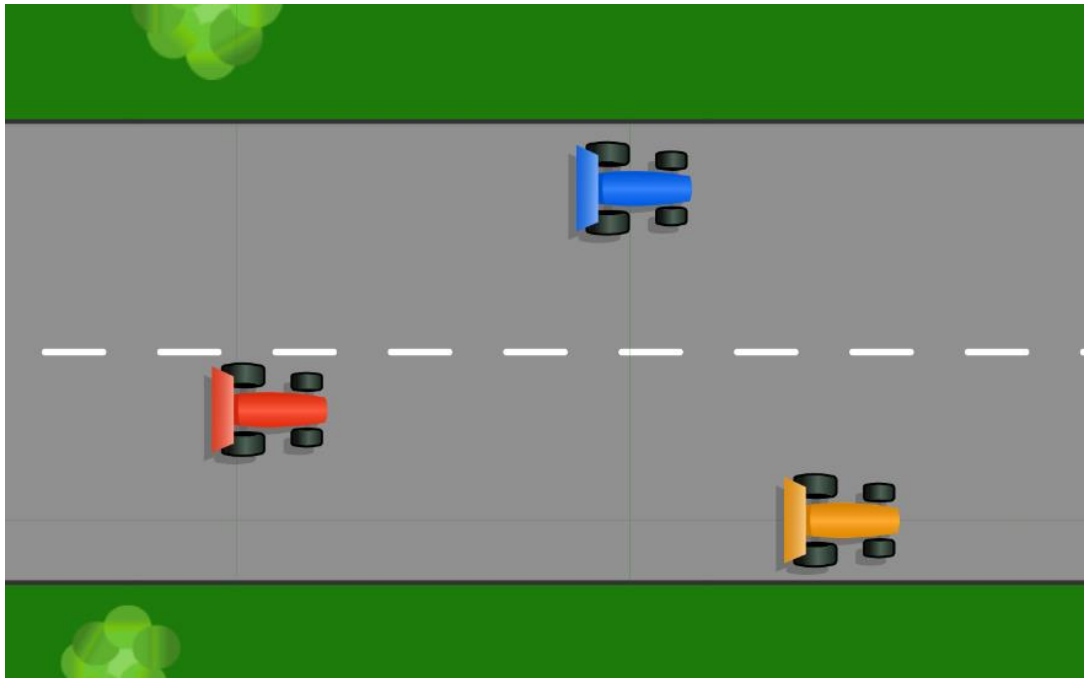
- *E.g. game of chess, reward is win/loss/tie only available at end of game*
- *E.g. Robot navigation, some instantaneous reward e.g. following a predefined trajectory, not falling into a ditch, but eventual reward only on reaching goal*

In such settings, usually there is a *terminal* or *goal* state

- *Entire or large bunch of reward given when reaching that state*
- *Period from start state to one such terminal state is called an* episode
- *We usually do not discount rewards in episodic tasks*

# RL for Playing Video Games





**States**: can be encoded as distance of our car from other cars and our current speed

More challenging to take image pixels as state and use a CNN to infer these distances first

**Actions**: move ↑/↓, speed ↑/↓, do nothing

**Rewards**: various forms can be offered

*Small instantaneous reward for surviving one frame*

*Small reward for overtaking a competitor car*

*Large penalty for crashing onto another car*

Since number of states is small here, various techniques are at least feasible here e.g. value iteration, policy iteration, Q learning

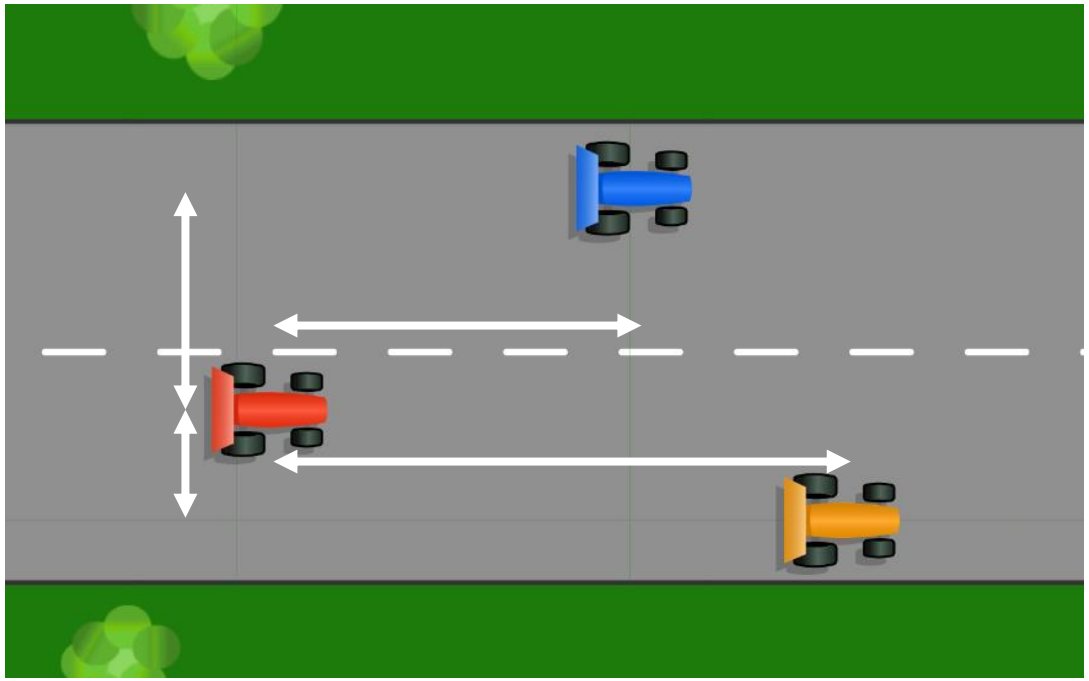
RL gained popularity for playing Atari games

HTTPS://WWW.ARCADEMICS.COM/GAMES/GRAND-PRIX

MNIH ET AL. PLAYING ATARI WITH DEEP REINFORCEMENT LEARNING, NIPS 2013

# RL for Playing Video Games





**States**: can be encoded as distance of our car from other cars and our current speed

More challenging to take image pixels as state and use a CNN to infer these distances first

**Actions**: move ↑/↓, speed ↑/↓, do nothing

**Rewards**: various forms can be offered

*Small instantaneous reward for surviving one frame*

*Small reward for overtaking a competitor car*

*Large penalty for crashing onto another car*

Since number of states is small here, various techniques are at least feasible here e.g. value iteration, policy iteration, Q learning

RL gained popularity for playing Atari games

HTTPS://WWW.ARCADEMICS.COM/GAMES/GRAND-PRIX

MNIH ET AL. PLAYING ATARI WITH DEEP REINFORCEMENT LEARNING, NIPS 2013

# RL for Adaptive Control



RL is widely used for controlling motors and other actuators in AE, ME applications

*Extremely challenging to fly a helicopter upside down*

*Required modifications to helicopter (8kgs) e.g. more powerful rotor, compass, gyroscope, accelerometer etc*

**States**: feedback from all these sensors

**Actions**: rotors controlling pitch (fw/bw, left/right), thrust, turning (yaw)

Human pilot was asked to demonstrate

*Was used to learn state transitions $P_{sa}$ i.e. the MDP*

**Rewards**: deviation of helicopter from set trajectory (Q learning used)

Several applications to robotics as well

NG ET AL, AUTONOMOUS INVERTED HELICOPTER FLIGHT VIA REINFORCEMENT LEARNING, EXPERIMENTAL ROBOTICS IX, 2006

# RL for Program Repair



```
1  #include<stdio.h>
2  int main(){
3    float ti, tax;
4    scanf ( "%f" ; &ti);
5    if(ti<200001){
6      printf("ti=0");}
7    else if(200000<ti && ti<500001){
8      tax=0.1*(ti-200000);
9      printf("%.2f", tax);}
10   else if(500000<ti && ti<1000001){
11     tax=30000+0.2*(ti-500000);
12     printf ( "%.2f" , tax ) ;
13   else if(ti>1000000){
14     tax=130000+0.3*(ti-1000000);
15     printf("%.2f", tax);}
16   return 0;}
```

$e_1$

$e_2$

**States**: encoding of program as string of tokens and posn of cursor within that string

**Actions**: two types of actions

- Navigation*: move the cursor around (these actions do not change the program at all)*
- Edit*: insert/delete/replace token at cursor posn*

**Rewards**: episodic learning

- *Full reward given when all errors removed*
- *Partial reward for removing one error*

*NN used to get low-dim representation of states*

*Linear models used to predict the best action by treating it as a multi-class problem*

GUPTA ET AL, DEEP REINFORCEMENT LEARNING FOR SYNTACTIC ERROR REPAIR IN STUDENT PROGRAMS, AAAI 2019

# RL for Planning and Routing



VALADARSKY ET AL, LEARNING TO ROUTE WITH DEEP RL, NIPS 2017

# RL for Planning and Routing



One of the earliest applications of RL

VALADARSKY ET AL, LEARNING TO ROUTE WITH DEEP RL, NIPS 2017

# RL for Planning and Routing



One of the earliest applications of RL

*Boyan and Littman, NIPS 93 – proposed Q learning*

VALADARSKY ET AL, LEARNING TO ROUTE WITH DEEP RL, NIPS 2017

# RL for Planning and Routing



One of the earliest applications of RL

*Boyan and Littman, NIPS 93 – proposed Q learning*

Recent works: delivery vehicle and network routing

VALADARSKY ET AL, LEARNING TO ROUTE WITH DEEP RL, NIPS 2017

# RL for Planning and Routing



One of the earliest applications of RL

*Boyan and Littman, NIPS 93 – proposed Q learning*

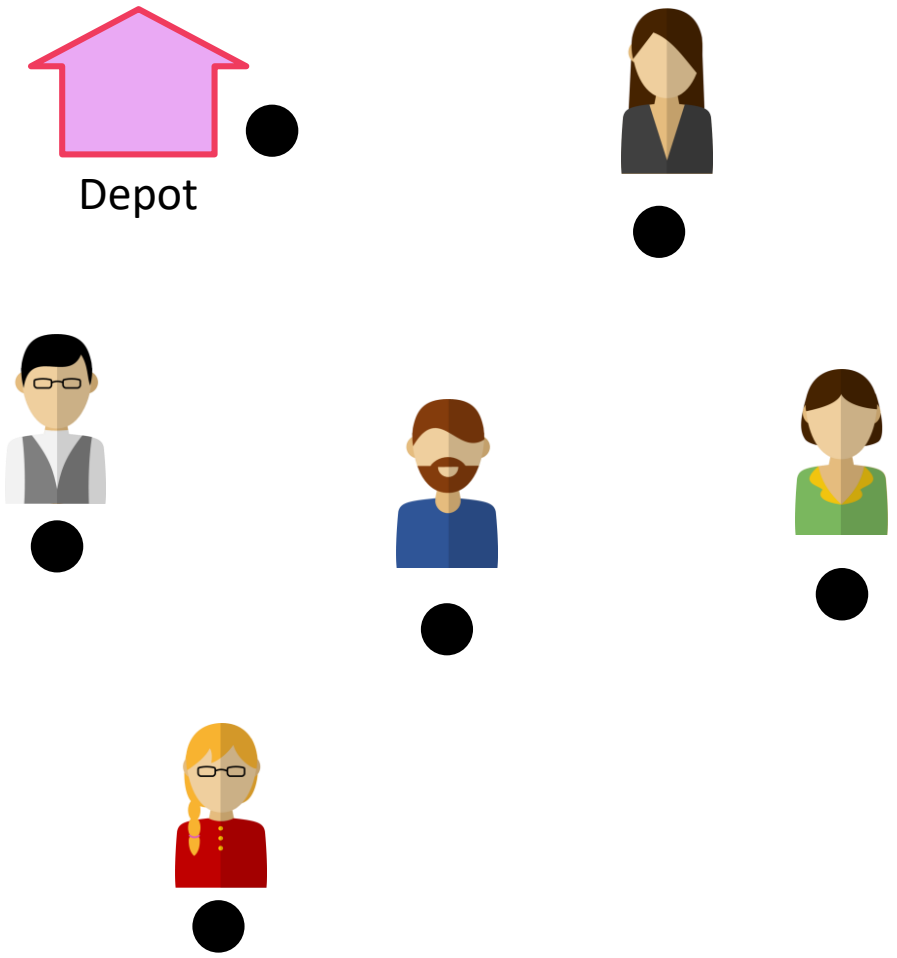
Recent works: delivery vehicle and network routing

Optimize route of delivery vehicle so that food/items can be delivered to maximum users in single round trip

VALADARSKY ET AL, LEARNING TO ROUTE WITH DEEP RL, NIPS 2017

# RL for Planning and Routing






One of the earliest applications of RL

*Boyan and Littman, NIPS 93 – proposed Q learning*

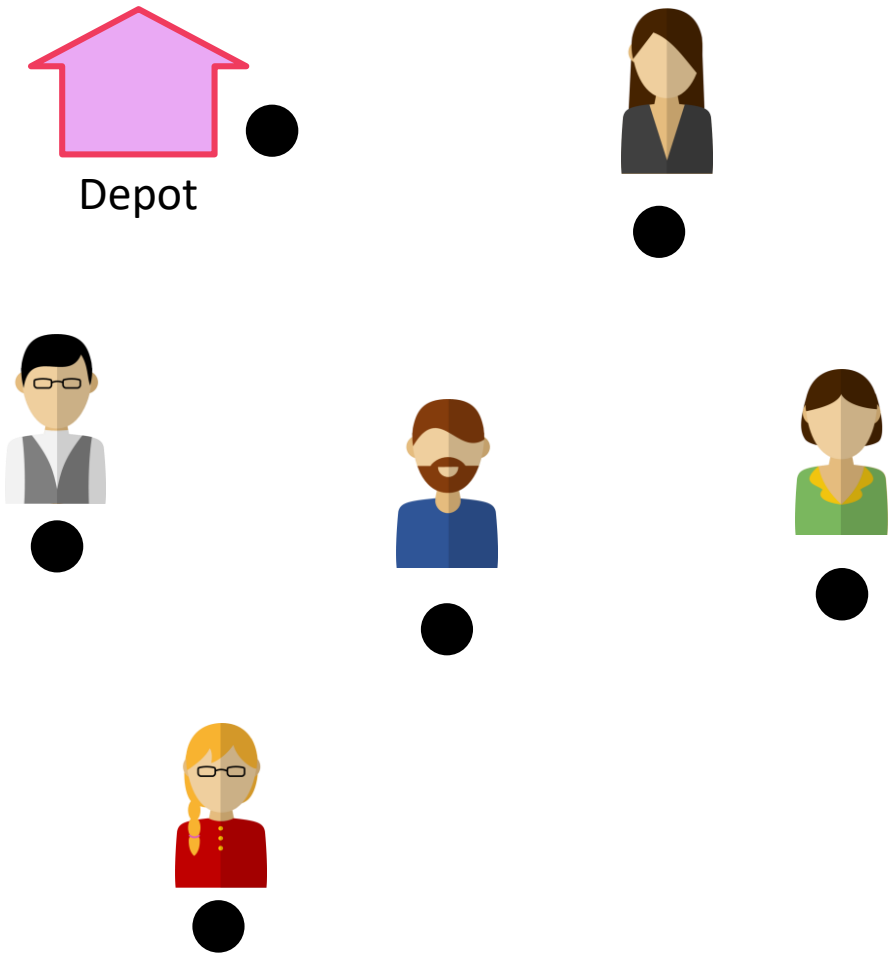
Recent works: delivery vehicle and network routing

Optimize route of delivery vehicle so that food/items can be delivered to maximum users in single round trip

VALADARSKY ET AL, LEARNING TO ROUTE WITH DEEP RL, NIPS 2017

# RL for Planning and Routing






One of the earliest applications of RL

*Boyan and Littman, NIPS 93 – proposed Q learning*

Recent works: delivery vehicle and network routing

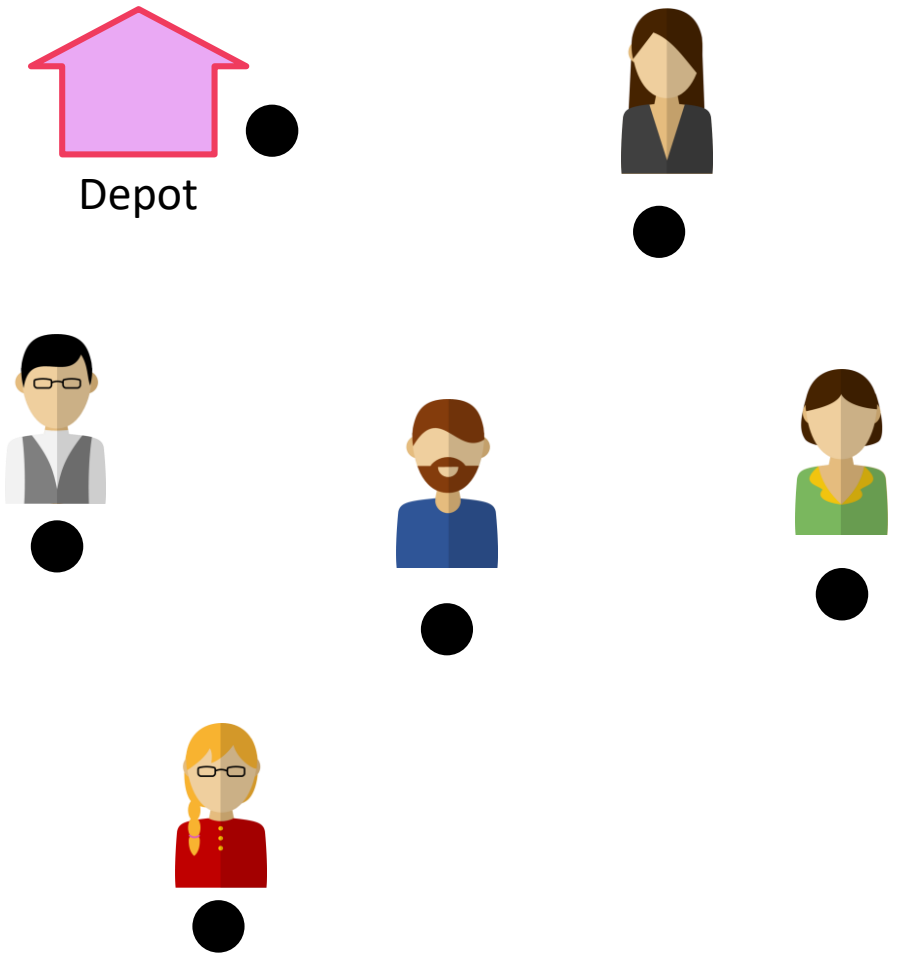
Optimize route of delivery vehicle so that food/items can be delivered to maximum users in single round trip

**States**: encoding of user locations, (remaining) demand for each user, (current) traffic conditions

VALADARSKY ET AL, LEARNING TO ROUTE WITH DEEP RL, NIPS 2017

# RL for Planning and Routing






One of the earliest applications of RL

*Boyan and Littman, NIPS 93 – proposed Q learning*

Recent works: delivery vehicle and network routing

Optimize route of delivery vehicle so that food/items can be delivered to maximum users in single round trip

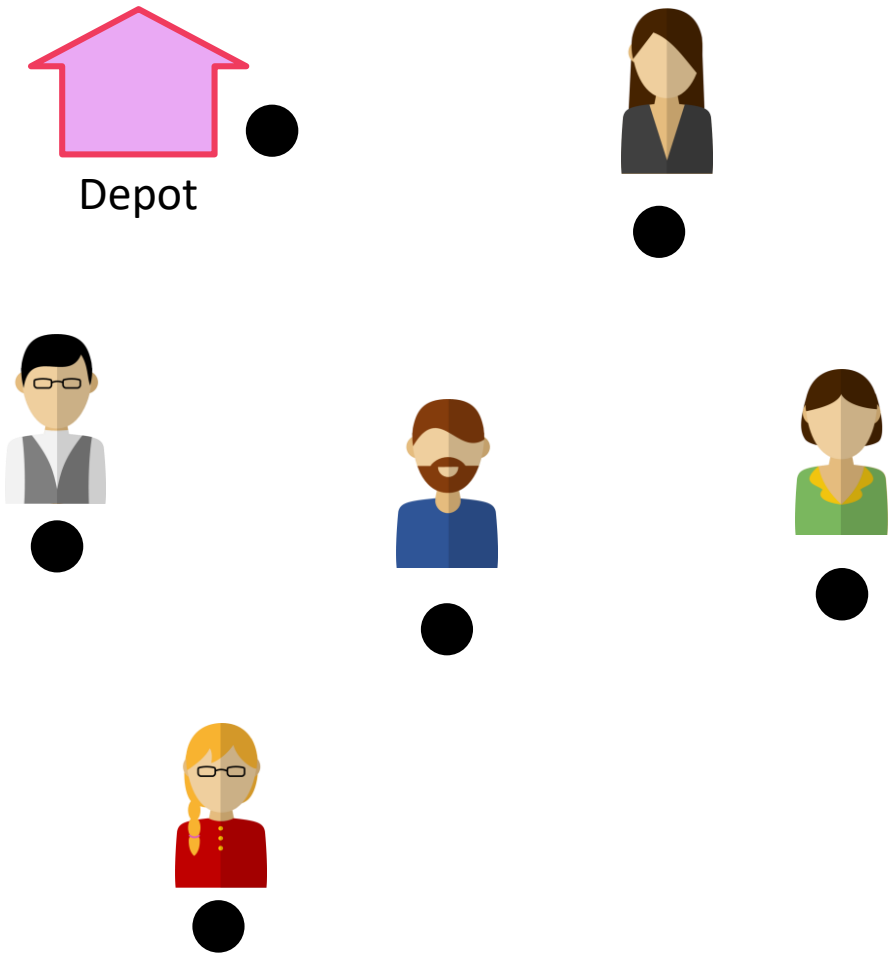
**States**: encoding of user locations, (remaining) demand for each user, (current) traffic conditions

*Our actions change the state by fulfilling (some) user demands but can be assumed to not change traffic conditions*

VALADARSKY ET AL, LEARNING TO ROUTE WITH DEEP RL, NIPS 2017

# RL for Planning and Routing






One of the earliest applications of RL

*Boyan and Littman, NIPS 93 – proposed Q learning*

Recent works: delivery vehicle and network routing

Optimize route of delivery vehicle so that food/items can be delivered to maximum users in single round trip

**States**: encoding of user locations, (remaining) demand for each user, (current) traffic conditions

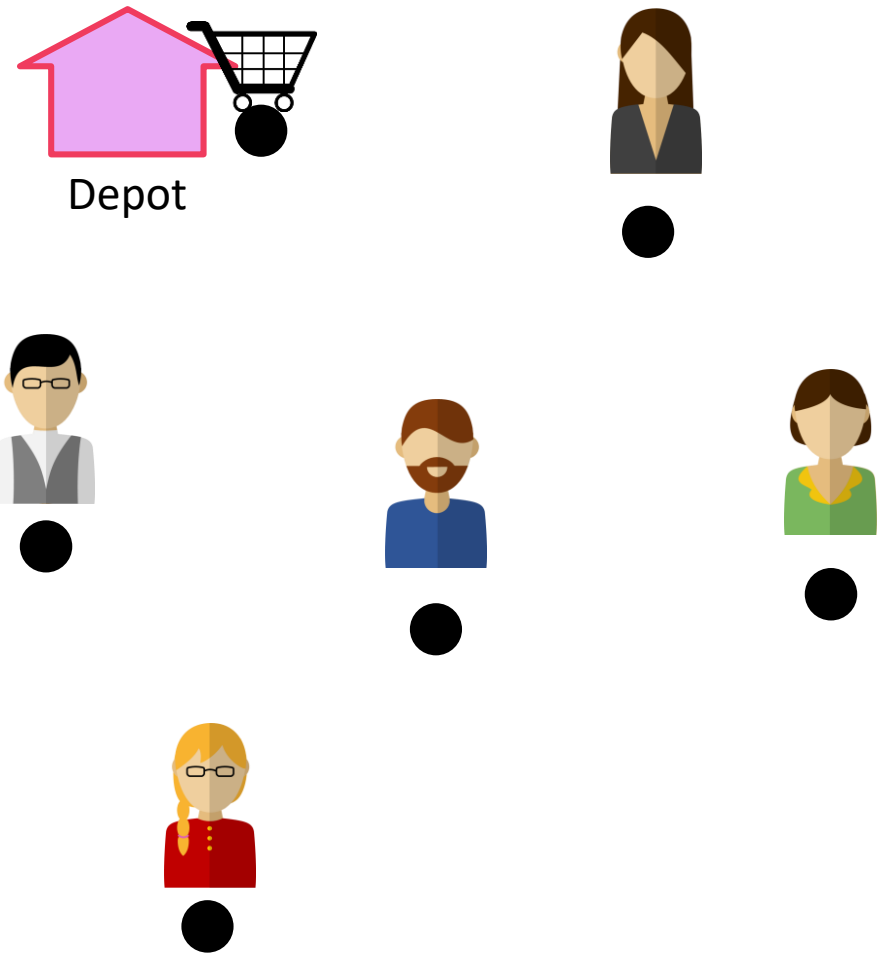
*Our actions change the state by fulfilling (some) user demands but can be assumed to not change traffic conditions*

**Actions**: given state, find next user to serve

VALADARSKY ET AL, LEARNING TO ROUTE WITH DEEP RL, NIPS 2017

# RL for Planning and Routing






One of the earliest applications of RL

*Boyan and Littman, NIPS 93 – proposed Q learning*

Recent works: delivery vehicle and network routing

Optimize route of delivery vehicle so that food/items can be delivered to maximum users in single round trip

**States**: encoding of user locations, (remaining) demand for each user, (current) traffic conditions

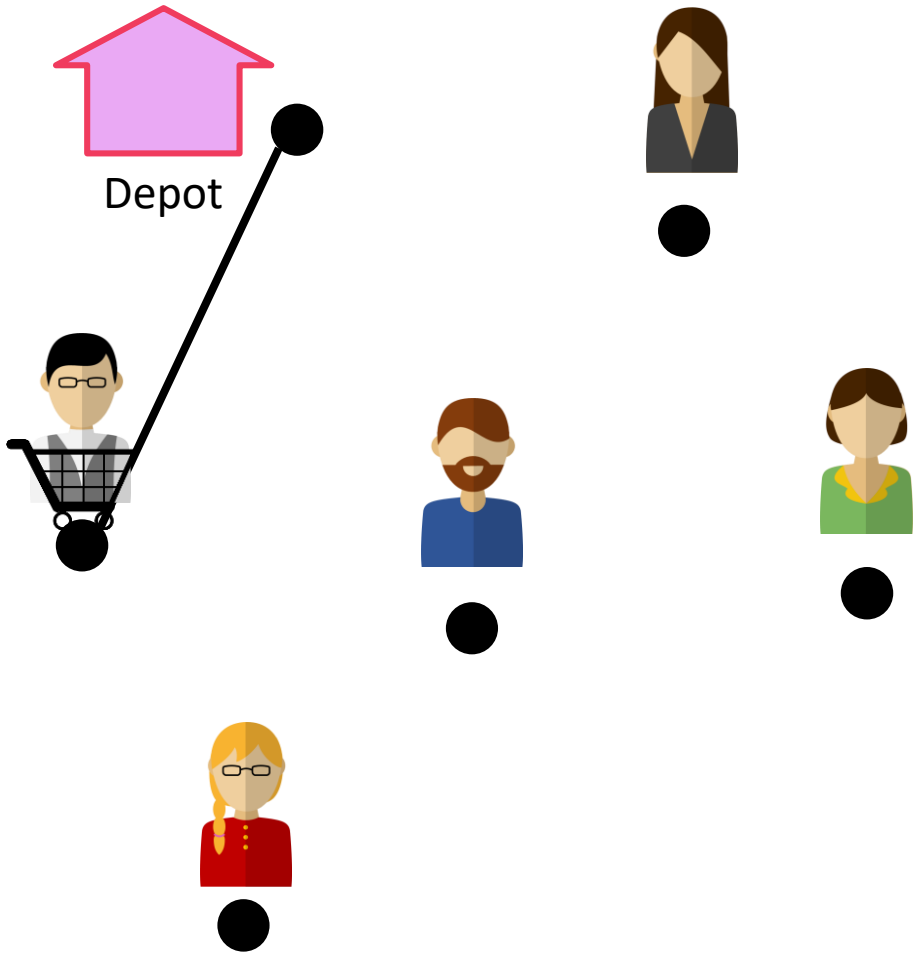
*Our actions change the state by fulfilling (some) user demands but can be assumed to not change traffic conditions*

**Actions**: given state, find next user to serve

VALADARSKY ET AL, LEARNING TO ROUTE WITH DEEP RL, NIPS 2017

# RL for Planning and Routing






One of the earliest applications of RL

*Boyan and Littman, NIPS 93 – proposed Q learning*

Recent works: delivery vehicle and network routing

Optimize route of delivery vehicle so that food/items can be delivered to maximum users in single round trip

**States**: encoding of user locations, (remaining) demand for each user, (current) traffic conditions

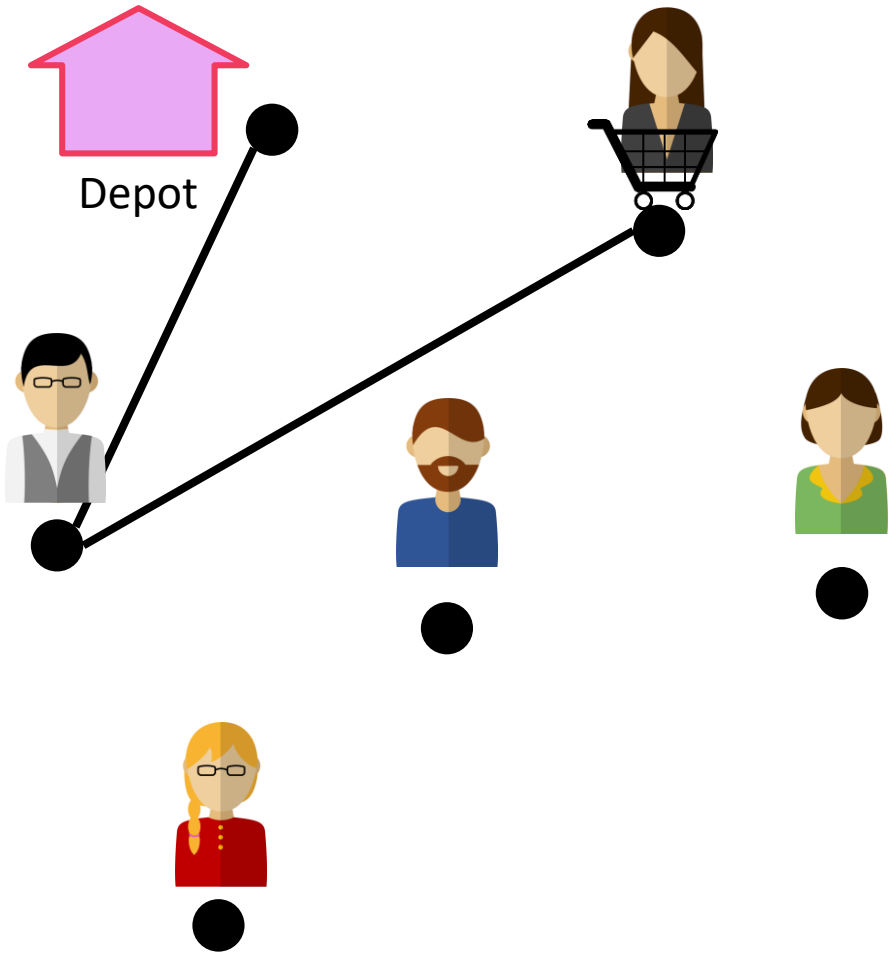
*Our actions change the state by fulfilling (some) user demands but can be assumed to not change traffic conditions*

**Actions**: given state, find next user to serve

VALADARSKY ET AL, LEARNING TO ROUTE WITH DEEP RL, NIPS 2017

# RL for Planning and Routing






One of the earliest applications of RL

*Boyan and Littman, NIPS 93 – proposed Q learning*

Recent works: delivery vehicle and network routing

Optimize route of delivery vehicle so that food/items can be delivered to maximum users in single round trip

**States**: encoding of user locations, (remaining) demand for each user, (current) traffic conditions

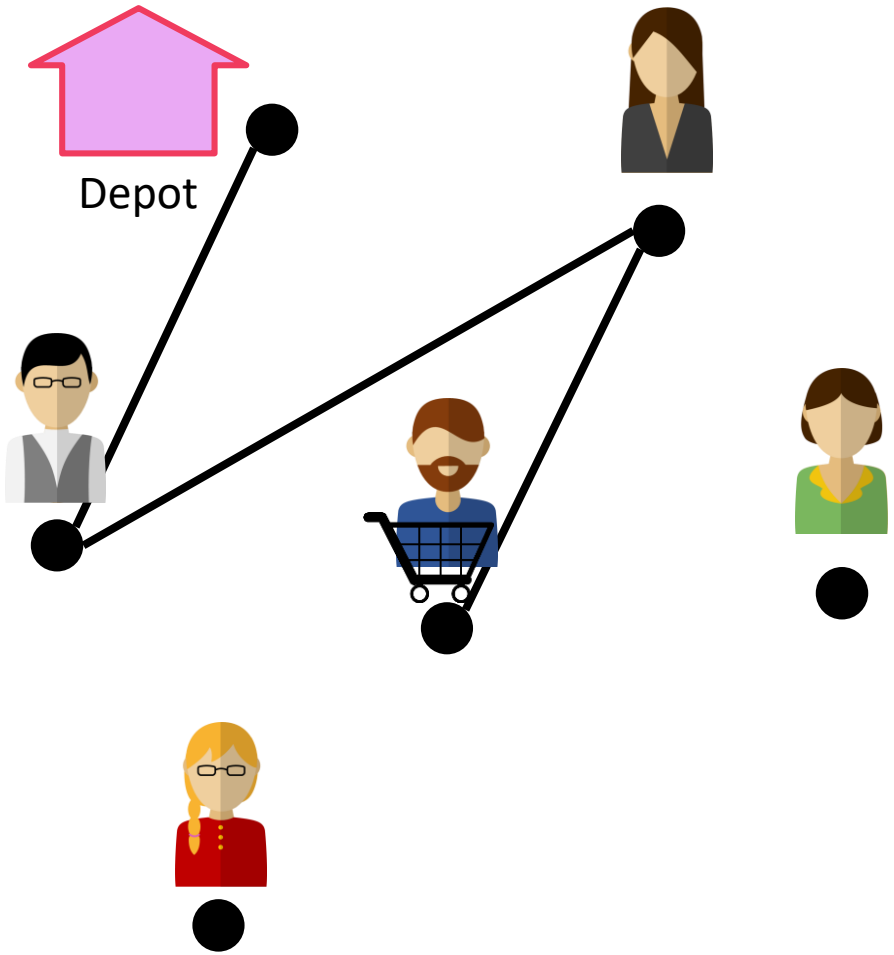
*Our actions change the state by fulfilling (some) user demands but can be assumed to not change traffic conditions*

**Actions**: given state, find next user to serve

VALADARSKY ET AL, LEARNING TO ROUTE WITH DEEP RL, NIPS 2017

# RL for Planning and Routing






One of the earliest applications of RL

*Boyan and Littman, NIPS 93 – proposed Q learning*

Recent works: delivery vehicle and network routing

Optimize route of delivery vehicle so that food/items can be delivered to maximum users in single round trip

**States**: encoding of user locations, (remaining) demand for each user, (current) traffic conditions

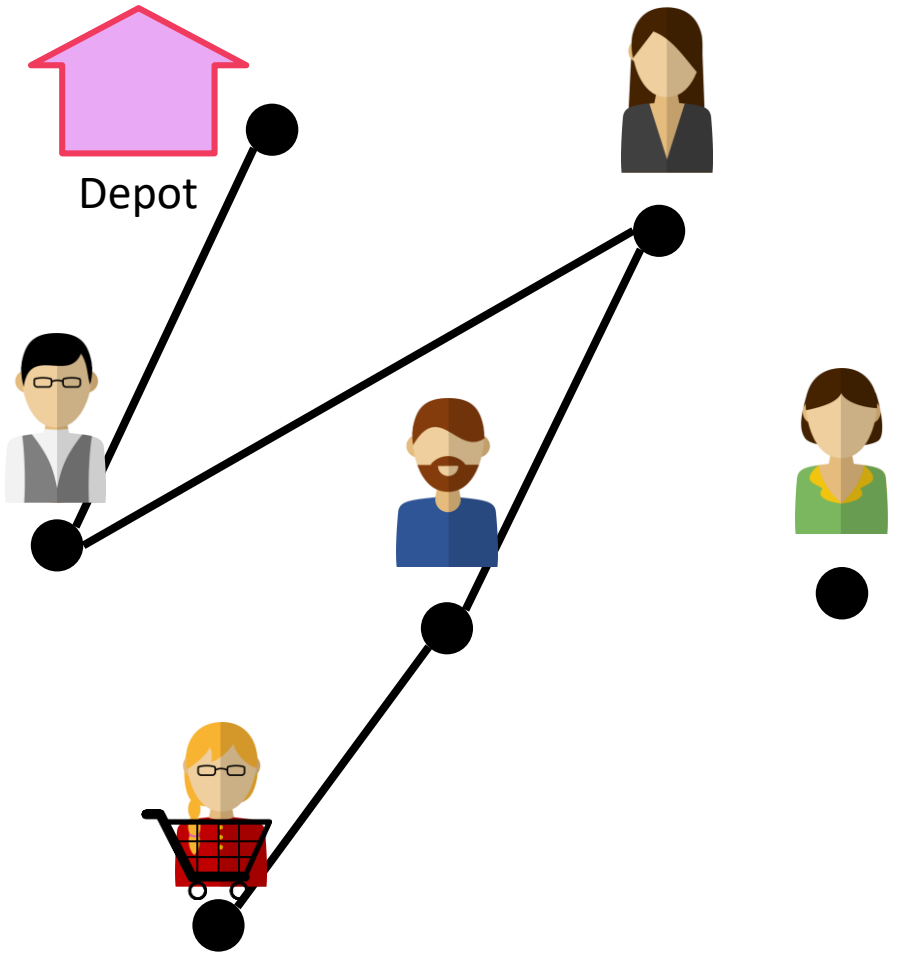
*Our actions change the state by fulfilling (some) user demands but can be assumed to not change traffic conditions*

**Actions**: given state, find next user to serve

VALADARSKY ET AL, LEARNING TO ROUTE WITH DEEP RL, NIPS 2017

# RL for Planning and Routing






One of the earliest applications of RL

*Boyan and Littman, NIPS 93 – proposed Q learning*

Recent works: delivery vehicle and network routing

Optimize route of delivery vehicle so that food/items can be delivered to maximum users in single round trip

**States**: encoding of user locations, (remaining) demand for each user, (current) traffic conditions

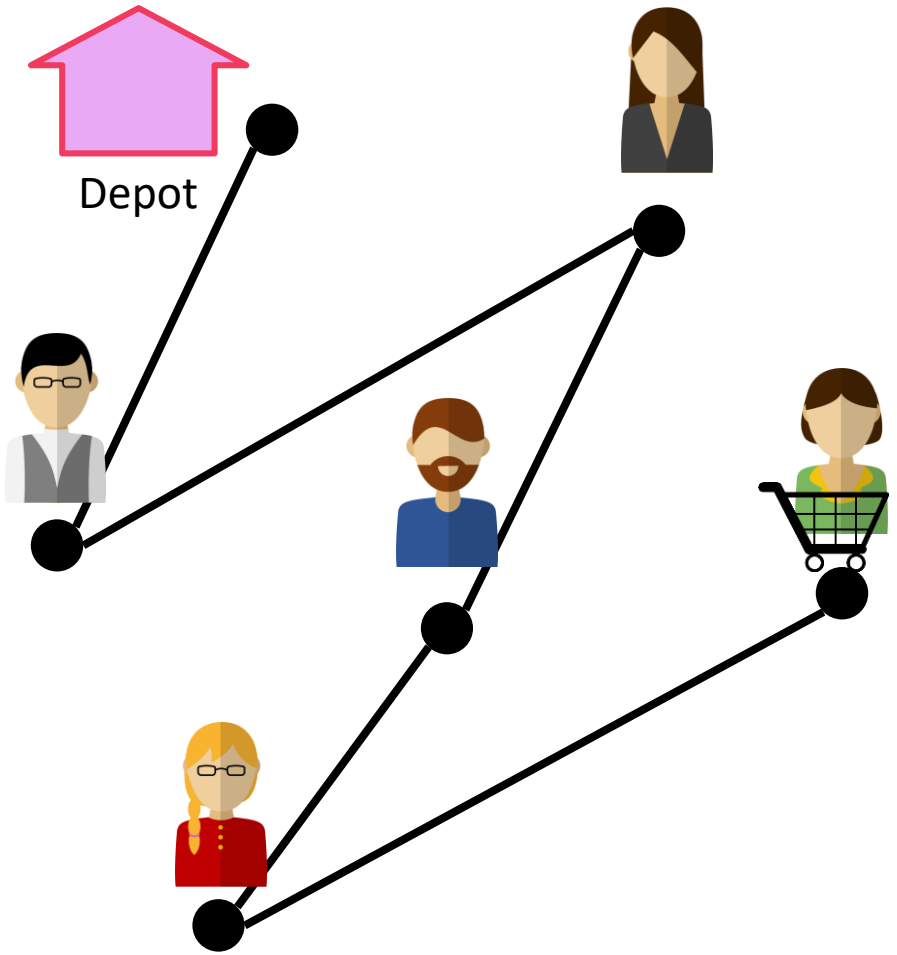
*Our actions change the state by fulfilling (some) user demands but can be assumed to not change traffic conditions*

**Actions**: given state, find next user to serve

VALADARSKY ET AL, LEARNING TO ROUTE WITH DEEP RL, NIPS 2017

# RL for Planning and Routing






One of the earliest applications of RL

*Boyan and Littman, NIPS 93 – proposed Q learning*

Recent works: delivery vehicle and network routing

Optimize route of delivery vehicle so that food/items can be delivered to maximum users in single round trip

**States**: encoding of user locations, (remaining) demand for each user, (current) traffic conditions

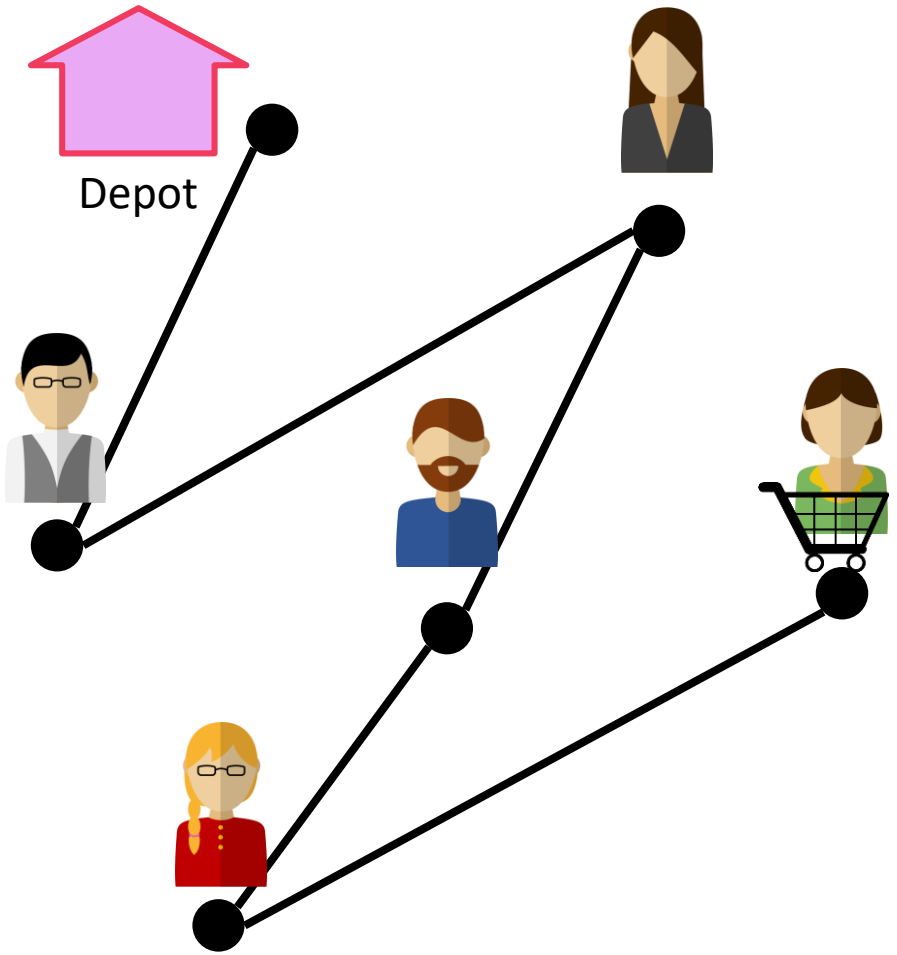
*Our actions change the state by fulfilling (some) user demands but can be assumed to not change traffic conditions*

**Actions**: given state, find next user to serve

VALADARSKY ET AL, LEARNING TO ROUTE WITH DEEP RL, NIPS 2017

# RL for Planning and Routing






One of the earliest applications of RL

*Boyan and Littman, NIPS 93 – proposed Q learning*

Recent works: delivery vehicle and network routing

Optimize route of delivery vehicle so that food/items can be delivered to maximum users in single round trip

**States**: encoding of user locations, (remaining) demand for each user, (current) traffic conditions

*Our actions change the state by fulfilling (some) user demands but can be assumed to not change traffic conditions*

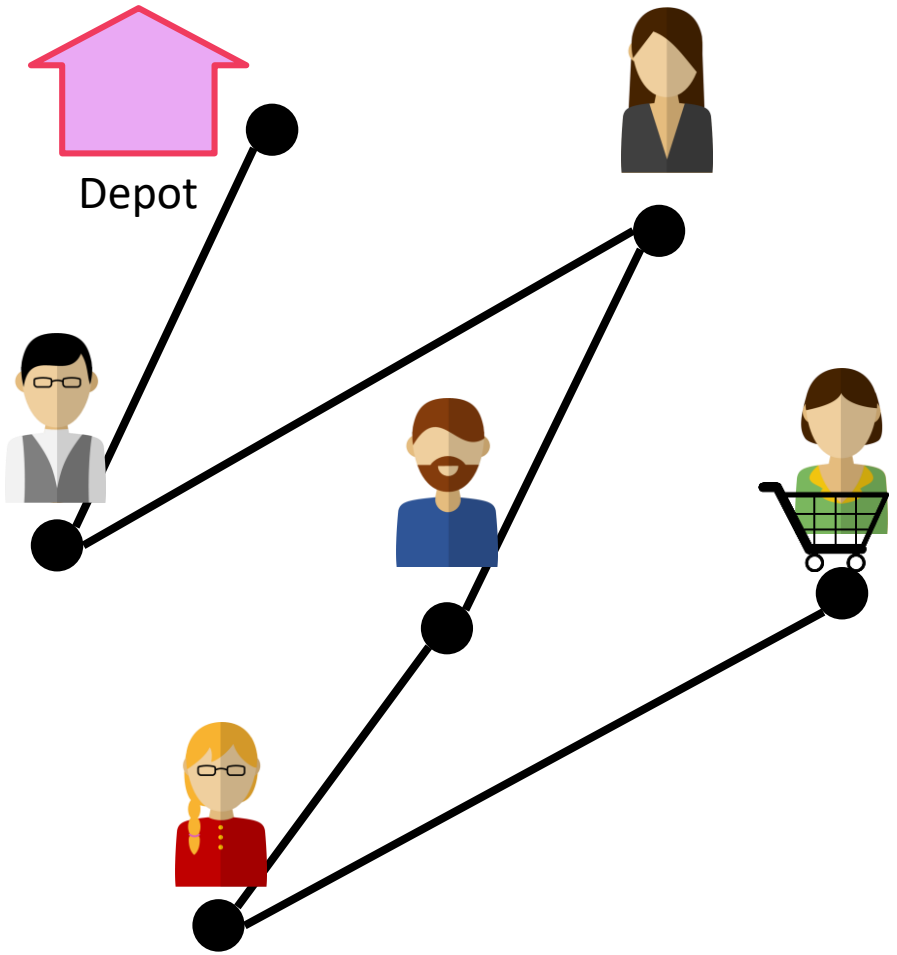
**Actions**: given state, find next user to serve

**Rewards**: instantaneous rewards

VALADARSKY ET AL, LEARNING TO ROUTE WITH DEEP RL, NIPS 2017

# RL for Planning and Routing






One of the earliest applications of RL

*Boyan and Littman, NIPS 93 – proposed Q learning*

Recent works: delivery vehicle and network routing

Optimize route of delivery vehicle so that food/items can be delivered to maximum users in single round trip

**States**: encoding of user locations, (remaining) demand for each user, (current) traffic conditions

*Our actions change the state by fulfilling (some) user demands but can be assumed to not change traffic conditions*

**Actions**: given state, find next user to serve

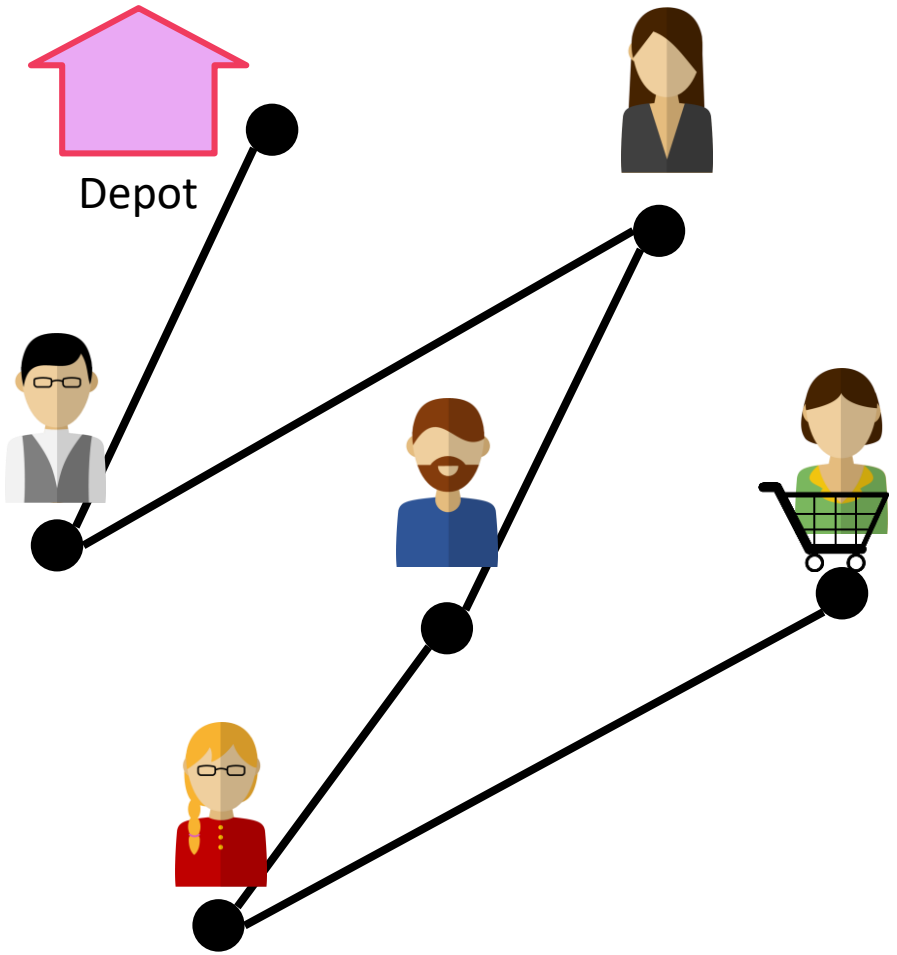
**Rewards**: instantaneous rewards

Positive reward for amount of demand fulfilled

VALADARSKY ET AL, LEARNING TO ROUTE WITH DEEP RL, NIPS 2017

# RL for Planning and Routing






One of the earliest applications of RL

*Boyan and Littman, NIPS 93 – proposed Q learning*

Recent works: delivery vehicle and network routing

Optimize route of delivery vehicle so that food/items can be delivered to maximum users in single round trip

**States**: encoding of user locations, (remaining) demand for each user, (current) traffic conditions

*Our actions change the state by fulfilling (some) user demands but can be assumed to not change traffic conditions*

**Actions**: given state, find next user to serve

**Rewards**: instantaneous rewards

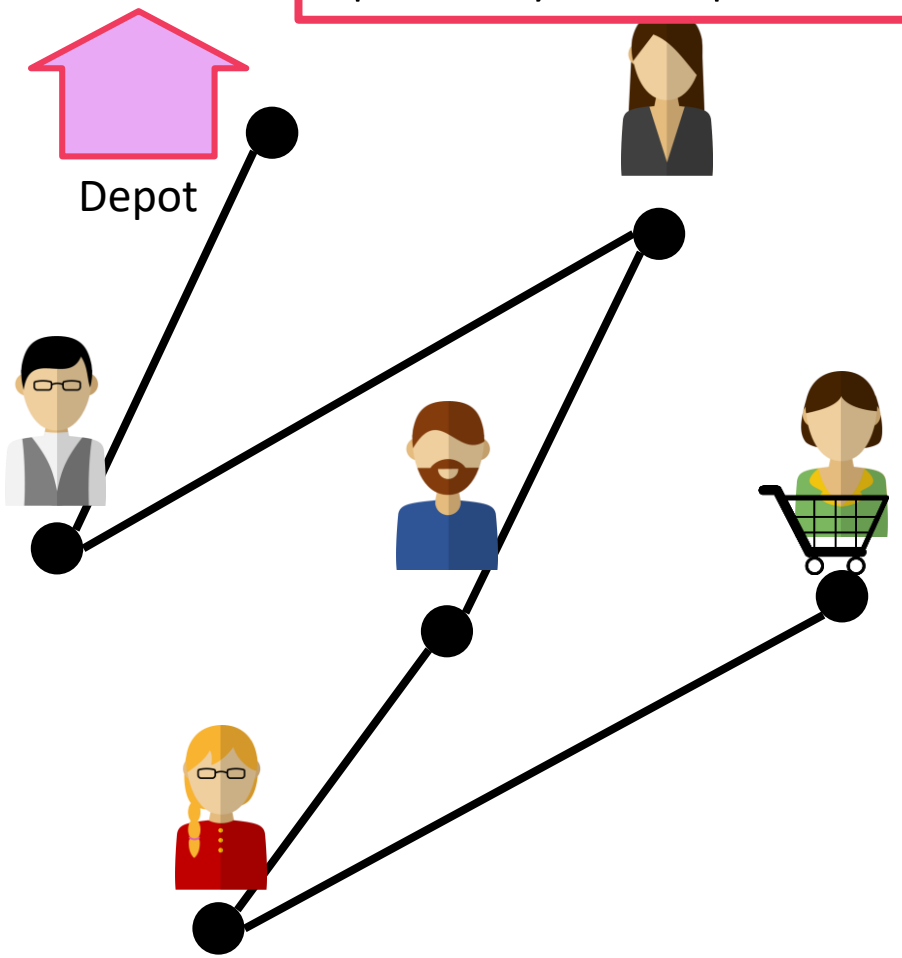
Positive reward for amount of demand fulfilled

Penalty for amount of time taken to do so (traffic etc)

VALADARSKY ET AL, LEARNING TO ROUTE WITH DEEP RL, NIPS 2017

# RL for

Numerous other applications in economics, finance, governance. Basically in any situation where "environment" changes dynamically, possibly in response to our actions, RL is a valuable candidate algo




One of the earliest applications of RL

*Boyan and Littman, NIPS 93 – proposed Q learning*

Recent works: delivery vehicle and network routing

Optimize route of delivery vehicle so that food/items can be delivered to maximum users in single round trip

**States**: encoding of user locations, (remaining) demand for each user, (current) traffic conditions

*Our actions change the state by fulfilling (some) user demands but can be assumed to not change traffic conditions*

**Actions**: given state, find next user to serve

**Rewards**: instantaneous rewards

Positive reward for amount of demand fulfilled

Penalty for amount of time taken to do so (traffic etc)

VALADARSKY ET AL, LEARNING TO ROUTE WITH DEEP RL, NIPS 2017

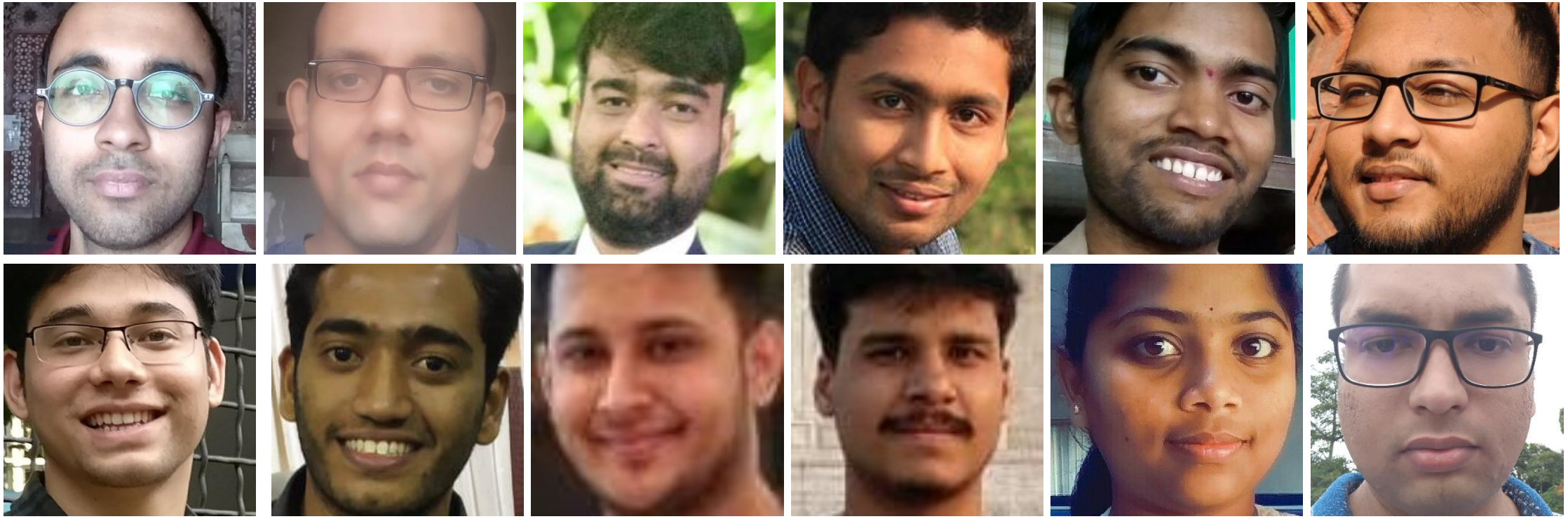
# Other Learning Paradigms

- Learning with incomplete/deficient/semi/active supervision
- Learning with *drift* (test data slightly different from train data)
- Learning multiple-tasks (generalization of multi-label learning)
- Learning with multiple views (several feature rep of same data pt)
- Learning with adversarial train/test data (robust ML)
- Learning under resource constraints (tiny/edge ML for IOT devices)
- Learning with online/streaming data
- Have seen glimpses of some of these in the course but only just ☺

# Acknowledgements



Course Team:



**CSE Office staff members**: Mr Prashant Kumar Sahu, Mr Rajesh Kumar, Mr Ranjan Kumar, Mr Anubhav Kumar Arya, Mr Amit Kumar Bharti

**Piazza, Gradescope** teams: invaluable for large courses such as ours