CE 273
Markov Decision Processes

Lecture 23

Inverse Reinforcement Learning

Previously on Markov Decision Processes

Discrete choice theory is built on the assumption that decision makers calculate the utility from different alternatives and choose the one with maximum utility.

But an analyst may not have access to many attributes that individuals consider when choosing an alternative. For instance, consider the problem of selecting a mode.

Decision maker's world:

- Cost
- Time
- Reliability
- Safety

Utility of a mode is some function f(Cost, Time, Reliability, Safety)

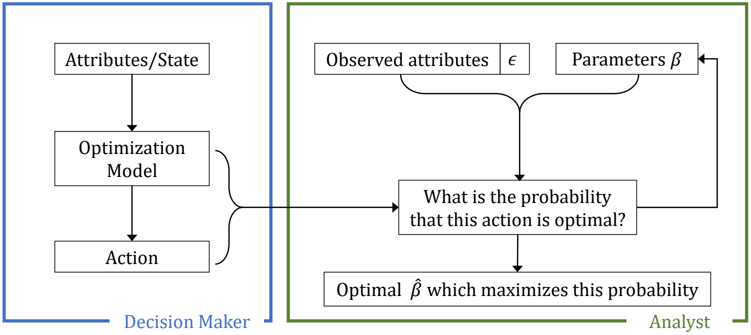
Analyst's world:

- Cost
- Time

Utility of a mode is some parametric function u(Cost, Time; β) + random component

The random component captures the effect of all unobserved or latent attributes such as reliability and safety which are difficult to measure.

Previously on Markov Decision Processes



Different optimization models maybe used in different contexts:

- Find the maximum of a bunch of numbers $\rightarrow$ Static choice models
- Knapsack problems $\rightarrow$ Multiple discrete-continuous choice models
- Markov Decision Process $\rightarrow$ Dynamic choice models

Previously on Markov Decision Processes

Just like static choice models, the expression for the probability can be written as follows

$$\mathbb{P}[a_t|x_t] = \mathbb{P}[v_t(x_t, a_t) + \epsilon_t(a_t) > v_t(x_t, a'_t) + \epsilon_t(a'_t) \, \forall \, a'_t \in A_t \backslash \{a_t\}]$$

where $v_t(x_t, a_t)$ is the **conditional value function**, which is a measure of the utility from choosing a_t at time t and behaving "optimally" thereafter.

Recall that the conditional value function was the utility of choosing an action and behaving optimally thereafter. Hence, we can write,

$$v_t(x_t, a_t) = u(x_t, a_t) + \alpha \int \bar{V}_{t+1}(x_{t+1}) f(x_{t+1}|x_t, a_t) dx_{t+1}$$

If ϵs are assumed to be Gumbel distributed, the above expectation has a closed form solution which is given by

$$\bar{V}_t(x_t) = \gamma + \ln \sum_{a_t \in A_t} \exp\left(v_t(x_t, a_t)\right)$$

Dynamic Choice Models

Maximum Likelihood Estimation

The one-step utilities are parameterized linearly using β's as done in static models, i.e., we can write $u(x_t, a_t)$ as $u(x_t, a_t; \beta)$.

Likewise, the transition beliefs are parameterized by λ. That is, we can write $f(x_{t+1}|x_t, a_t)$ as $f(x_{t+1}|x_t, a_t; \lambda)$.

Finally, we can use the time-dependent choice probabilities to define a likelihood function and maximize it to obtain estimates of the discount factor and utility functions (and also the transition functions)

$$\mathbb{P}[a_{nt}|x_{nt}; \alpha, \beta, \lambda] = \frac{\exp\left(v_{nt}(x_{nt}, a_{nt}; \alpha, \beta, \lambda)\right)}{\sum_{a'_{nt} \in A_{nt}} \exp\left(v_{nt}(x_{nt}, a'_{nt}; \alpha, \beta, \lambda)\right)}$$

where n denotes an observation. The objective of the estimation procedure is to find $(\hat{\alpha}, \hat{\beta}, \hat{\lambda})$ which is a solution to

$$\max_{\alpha \in [0,1]} \mathcal{LL}(\alpha, \beta, \lambda) = \sum_{n=1}^{N} \sum_{t=1}^{T} \ln\left(\mathbb{P}[a_{nt}|x_{nt}; \alpha, \beta, \lambda]\right)$$

Lecture Outline

Lecture Outline

Inverse Reinforcement Learning

Inverse Reinforcement Learning

Introduction

Inverse RL like dynamic discrete choice models also attempts to recover the problem data by observing actions taken by an agent, assuming that the agent was behaving optimally.

We will assume that agents maximize rewards.

The utilities/rewards are also linearly parametrized using a set of features. However, there are a few key differences:

- The rewards are assumed deterministic and there are no latent components unlike random utility models.
- Thus, the optimization formulations involved are not stochastic.
- Beliefs on transition probabilities are not usually modeled and we assume that we have access to the environment to test any policy.

Inverse Reinforcement Learning

Introduction

Assume that an expert demonstrates his/her actions in an environment using a policy μ^*. The objective is to find the one-stage rewards $g(x, u)$ that the agent had in mind.

In reality, we only have sample paths taken by the agent and may not be able to access the full policy. We make this assumption only to keep the notation simple. Methods used in practice rely only on trajectories taken by agents.

For similar reasons, we will also assume that the one-stage rewards are independent of the action taken and hence try to find $g(x)$ instead.

Inverse Reinforcement Learning

Overview

The idea behind inverse RL is simple. Since μ^* is optimal according to the expert, the reward from using it should be greater than that obtained from any other policy.

This condition can be translated to a bunch of inequality conditions. The problem however is that there are many reward functions which satisfy it.

Inverse Reinforcement Learning

Overview

Mathematically, suppose μ^* is the policy used by an expert and μ is some admissible stationary policy. Then, from Bellman equations,

$$J_\mu = T_\mu J_\mu = g + P_\mu J_\mu$$

Thus, J_μ can also be written as $(I - P_\mu)^{-1}g$. Note that since g does not depend on u, the subscript μ in g_μ can be suppressed.

For μ^* to be optimal, we must therefore have

$$(I - P_{\mu^*})^{-1}g \geq (I - P_\mu)^{-1}g \ \forall \ \mu \in \Pi$$

What is an obvious reward vector which satisfies the above inequalities?

Inverse Reinforcement Learning

Overview

In order to address the uniqueness issue, a secondary objective is imposed. Different studies have used different objectives depending on the application. Examples include

- Maximum margin methods
  - Ng, A. Y., & Russell, S. J. (2000, June). Algorithms for inverse reinforcement learning. In Proceedings of the 21 International Conference on Machine Learning (pp. 663-670).
  - Ratliff, N. D., Bagnell, J. A., & Zinkevich, M. A. (2006, June). Maximum margin planning. In Proceedings of the 23rd International Conference on Machine Learning (pp. 729-736).
- Entropy maximization
  - Ziebart, B. D., Maas, A. L., Bagnell, J. A., & Dey, A. K. (2008, July). Maximum Entropy Inverse Reinforcement Learning. In AAAI (Vol. 8, pp. 1433-1438).

Inverse Reinforcement Learning

Maximum Margin Methods

The objective in the maximum margin methods is choose an objective such that the rewards we discover will make μ^* a clear winner.

For instance, one option is to maximize the gap between the performance of μ^* and the second best policy.

For notational convenience, suppose that the process always starts in a state i and the expert is optimizing $J_{\mu^*}(i)$. Then, we can write the above problem as

$$\begin{aligned} &\max_{g} \min_{\mu} \Big(J_{\mu^*}(i) - J_{\mu}(i) \Big) \\ \text{s.t. } &(I - P_{\mu^*})^{-1} g(i) \geq (I - P_{\mu})^{-1} g(i) \qquad\qquad \forall\, \mu \in \Pi \end{aligned}$$

This is a linear program. Why?

Inverse Reinforcement Learning

Maximum Margin Methods

The problem can be rewritten as

$$\begin{aligned} &\max_{g}\ y \\ \text{s.t. } &(I - P_{\mu^*})^{-1} g(i) \geq (I - P_{\mu})^{-1} g(i) && \forall\, \mu \in \Pi \\ &y \leq J_{\mu^*}(i) - J_{\mu}(i) && \forall\, \mu \in \Pi \end{aligned}$$

In addition, one can also impose bounds on the magnitudes of the one-step rewards and also include a penalty term for having large rewards.

The idea behind these additional constraints is to keep the rewards "simple" and they have been found to perform better in practice.

Inverse Reinforcement Learning

Maximum Margin Methods

The LP formulation with the new constraints is written as

$$
\begin{aligned}
&\max_{g}\; y - \lambda \|g\|_1 \\
\text{s.t. }\; &(I - P_{\mu^*})^{-1} g \geq (I - P_{\mu})^{-1} g && \forall\, \mu \in \Pi \\
&y \leq J_{\mu^*}(i) - J_{\mu}(i) && \forall\, \mu \in \Pi \\
&|g(x)| \leq M && \forall\, x \in X
\end{aligned}
$$

The λ value is a parameter which balances the two objectives: separating the optimal policy from the second best solution and keeping the rewards small.

Are there any disadvantages of using the above LP for finding the rewards?

- We can have a large number of variables due to a large state space.
- The number of constraints can exponential since we have an inequality for every admissible policy μ.

We will address the first using a function approximation and the second using a simulation/column generation-type approach.

Inverse Reinforcement Learning

Function Approximation of Rewards

To overcome the issue of large state spaces, we assume that the the reward functions can be represented using a linear architecture similar to approximations in value space.

Define $g(x) = \phi(x)^\mathsf{T} w$ where $\phi(x) \in \mathbb{R}^m$ is a feature vector.The goal is now to find the optimal w^* which has fewer variables than the original g vector. Assume that there are n states. As before, define

$$\Phi = \begin{bmatrix} \phi_1(1) & \dots & \phi_m(1) \\ \phi_1(2) & \dots & \phi_m(2) \\ \vdots & \vdots & \vdots \\ \phi_1(n) & \dots & \phi_m(n) \end{bmatrix}_{n \times m}$$

Then the one-step rewards can be written as

$$g = \Phi w$$

Inverse Reinforcement Learning

Function Approximation of Rewards

Since the one-step rewards are assumed independent of the actions, the value functions can be written as

$$J_\mu = g^\mathsf{T} \vartheta_\mu$$

where ϑ_μ is the discounted limiting distribution over the state space on the Markov chain induced by μ. Mathematically,

$$\vartheta_\mu(x) = \sum_{k=0}^{\infty} \alpha^k \mathbb{P}_\mu[x_k = x | x_0 = i]$$

If $\alpha = 1$, this is simply the limiting distribution.

Inverse Reinforcement Learning

Function Approximation of Rewards

Recall that the constraint $J_{\mu^*}(i) \geq J_\mu(i)$ in the earlier LP ensured that μ^* is optimal.

Writing J in terms of the discounted limiting distribution and using the linear architecture for g's,

$$\begin{aligned} & g^\mathsf{T}\vartheta_{\mu^*} \geq g^\mathsf{T}\vartheta_\mu \\ \Rightarrow & (\Phi w)^\mathsf{T}\vartheta_{\mu^*} \geq (\Phi w)^\mathsf{T}\vartheta_\mu \\ \Rightarrow & w^\mathsf{T}\Phi^\mathsf{T}\vartheta_{\mu^*} \geq w^\mathsf{T}\Phi^\mathsf{T}\vartheta_\mu \end{aligned}$$

The term $\Phi^\mathsf{T}\vartheta_\mu \in \mathbb{R}^m$ is called the **feature expectation** vector. Let us denote it using φ_μ. Therefore, the above expression becomes

$$w^\mathsf{T}\varphi_{\mu^*} \geq w^\mathsf{T}\varphi_\mu$$

Inverse Reinforcement Learning

Feature Expectations

The reason φ_μ is called the feature expectation is that the value function can be written as $J_\mu(i) = w^\mathsf{T}\varphi_\mu$. Expanding the LHS,

$$\begin{aligned} J_\mu(i) &= \mathbb{E}\left\{ \sum_{k=0}^{\infty} \alpha^k g(x_k) | x_0 = i, \mu \right\} \\ &= \mathbb{E}\left\{ \sum_{k=0}^{\infty} \alpha^k w^\mathsf{T} \phi(x_k) | x_0 = i, \mu \right\} \\ &= w^\mathsf{T} \mathbb{E}\left\{ \sum_{k=0}^{\infty} \alpha^k \phi(x_k) | x_0 = i, \mu \right\} \end{aligned}$$

Hence, φ_μ is the expectation of the discounted feature vectors.

$$\varphi_\mu = \mathbb{E}\left\{ \sum_{k=0}^{\infty} \alpha^k \phi(x) | x_0 = i, \mu \right\}$$

Inverse Reinforcement Learning

Function Approximation of Rewards

Recall that the original LP was

$$\begin{aligned} &\max_g \; y \\ \text{s.t. } &(I - P_{\mu^*})^{-1} g(i) \geq (I - P_\mu)^{-1} g(i) && \forall \mu \in \Pi \\ &y \leq J_{\mu^*}(i) - J_\mu(i) && \forall \mu \in \Pi \end{aligned}$$

Using the function approximation of g, we can thus reformulate it as

$$\begin{aligned} &\max_w \; y \\ \text{s.t. } &(I - P_{\mu^*})^{-1} \phi(i)^\mathsf{T} w \geq (I - P_\mu)^{-1} \phi(i)^\mathsf{T} w && \forall \mu \in \Pi \\ &y \leq w^\mathsf{T} \varphi_{\mu^*} - w^\mathsf{T} \varphi_\mu && \forall \mu \in \Pi \end{aligned}$$

How many variables and constraints are present in the new LP?

Inverse Reinforcement Learning

Function Approximation of Rewards

In practice, we cannot construct the constraint set using all policies. Hence, we can sample from the set of admissible policies.

Another option is to generate them iteratively:

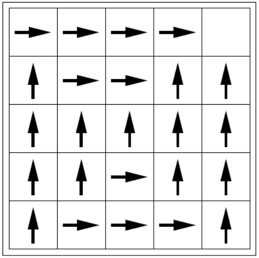
- Suppose we solve the LP by constructing constraints generated by the set of policies $\{\mu_1, \mu_2, \ldots, \mu_k\}$. Let the optimal weight vector be w_k^*.
- Resolve the MDP with a reward function $g(x) = \phi(x)^{\mathsf{T}} w_k^*$ and generate a new policy μ_{k+1} and repeat.

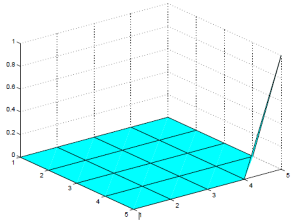
Defining a convergence criteria can be challenging. We will discuss one possible option later.

Inverse Reinforcement Learning

Example

Consider the following grid world example in which the objective is to go to the top right corner. In each step, one can move N, S, E, or W if possible but with a 30% chance they end up in a random direction.



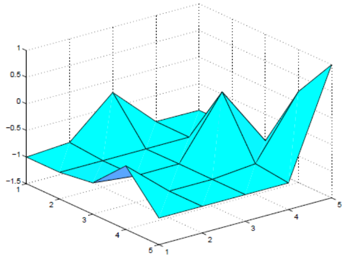


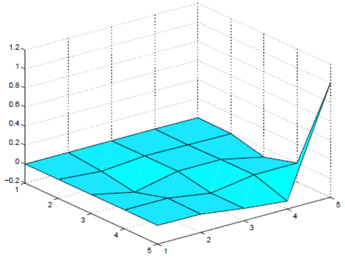
The optimal policy and the cost function is shown above.

Inverse Reinforcement Learning

Example

The results of the LP for $\lambda = 0$ (left) and $\lambda = 1.05$ (right) are shown below.





Inverse Reinforcement Learning

Alternate Formulations

There are other formulations for inverse RL which maximize the margin between the best and the other policies but the extent of separation is dependent on how different the two policies are. For instance,

$$
\begin{aligned}
&\min_{g,\xi} \|g\|_2^2 + \lambda \xi \\
\text{s.t. } & J_{\mu^*} \geq J_\mu + \mathcal{L}(\mu, \mu^*) - \xi \qquad\qquad \forall\, \mu \in \Pi
\end{aligned}
$$

The function $\mathcal{L}(\mu, \mu^*)$ is called the loss function and could be the number of states in which policies μ and μ^* prescribe different actions.

Inverse Reinforcement Learning

Alternate Formulations

We can parameterize the one-step rewards and also use a sampled set of constraints as done earlier.

However, note that the objective is quadratic, convex, but not differentiable.Subgradient methods have been found to perform better than off-the-shelf quadratic programming solvers.

Lecture Outline

Apprenticeship Learning

Apprenticeship Learning

Introduction

Inverse RL can be used in a supervised/imitation learning setting to mimic the behavior of an expert.

For instance, an expert may demonstrate how to drive or lift an object and a robot can first learn the one-step rewards of the expert and then solve an MDP using these rewards.

In these applications, it is more important to be right about the performance of the policy than the actual reward functions.

Apprenticeship Learning

Assumptions

We suppose that the true reward function of the expert is linear in the feature vector and is given by

$$g^*(x) = \phi(x)^\mathsf{T} w^*$$

where $w^* \in \mathbb{R}^m$ and $\|w^*\|_1 \leq 1$. As before, assume an initial state i for every problem instance.

The objective is to find a policy μ' that is similar to the one used by the expert μ^*. Recall that the performance of a policy μ, $J_\mu(i) = w^\mathsf{T}\varphi_\mu$. If the two policies have similar performance,

$$w^\mathsf{T}\varphi_{\mu'} \approx w^\mathsf{T}\varphi_{\mu^*}$$

Thus, if we can match the feature expectation vectors, we can be assured that our performance will match that of the expert. The algorithm that we will discuss will find a μ' such that $|w^\mathsf{T}\varphi_{\mu'} - w^\mathsf{T}\varphi_{\mu^*}| \leq \epsilon$.

Apprenticeship Learning

Estimating Feature Expectation Vectors

Recall that give a policy μ, its expected feature vector is defined as

$$\varphi_\mu = \mathbb{E}\left\{ \sum_{k=0}^{\infty} \alpha^k \phi(x) | x_0 = i, \mu \right\}$$

In practice, this expectation is estimated using samples of trajectories demonstrated by the expert or by a simulator.

Apprenticeship Learning

Algorithm

The main steps of the algorithm are:

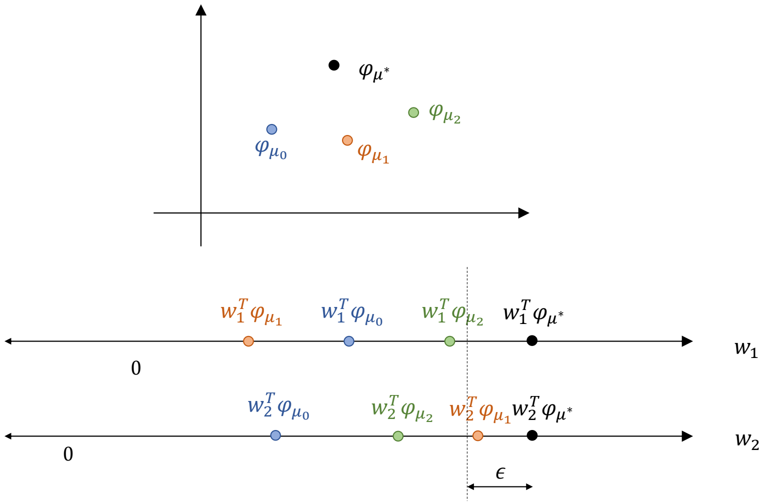
1. Pick some policy μ_0. Set $k = 1$
2. Solve a inverse RL quadratic programming problem

$$
\begin{aligned}
& y_k = \max_{y,w} y \\
\text{s.t.}\ & w^\mathsf{T}\varphi_{\mu^*} \geq w^\mathsf{T}\varphi_{\mu_j} + y \qquad\qquad \forall j = 0, 1, \ldots k-1 \\
& \|w\|_2 \leq 1
\end{aligned}
$$

3. If $y_k \leq \epsilon$, terminate.
4. Else, compute a new policy μ_k that is optimal for the MDP using rewards $g = \phi(x)^\mathsf{T} w_k$.
5. Estimate φ_k using simulation.
6. $k \leftarrow k + 1$ and go to Step 2.

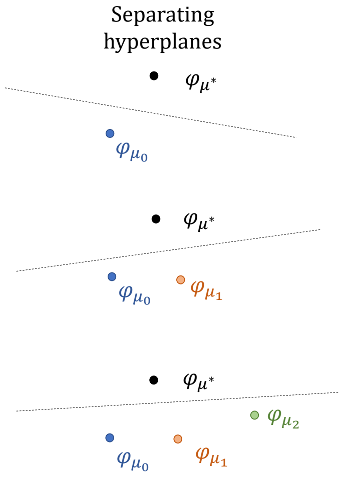
Apprenticeship Learning

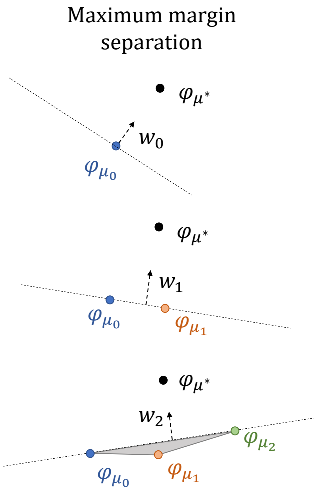
Algorithm



Apprenticeship Learning

Algorithm





Apprenticeship Learning

Algorithm

Upon convergence, the algorithm returns a set of policies $\mu_0, \mu_1, \ldots, \mu_K$.

As seen in the geometric interpretation, the performance of one of these policies is close to that used by the expert.

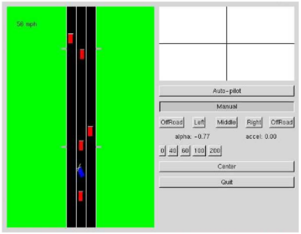
One can use the dual variables of the quadratic program to find this policy or define a new quadratic program on the convex closure of $\mu_0, \mu_1, \ldots, \mu_K$.

$$\begin{aligned} & \min \|\mu^* - \mu\|_2 \\ \text{s.t. } & \mu = \sum_k \lambda_k \mu_k \\ & \sum_k \lambda_k = 1 \\ & \lambda_k \geq 0 & \forall\, k = 0, 1, \ldots K \end{aligned}$$

The policy we will finally use will be a mixture in which we use policy μ_i with probability λ_i^* (optimal solution of the above problem).

Apprenticeship Learning
Example

This approach has successfully used to mimic driving behaviors on simple simulators.



The speed of the blue car is assumed to be fixed at 56 mph and is faster than all the red cars. One can choose to drive on one of the three lanes and two shoulders (green portions).

The feature vector includes the lane in which the car is currently in, the left shoulder, right shoulder, distance of the closest car in the current lane.

Apprenticeship Learning
Example

The some driving behaviors demonstrated by the expert include

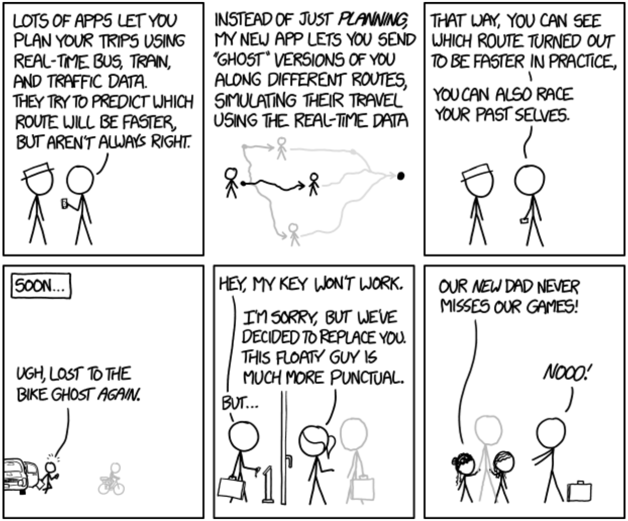
- Nice: Avoid collisions and shoulders and prefer driving on the right
- Nasty: Hit as many cars as possible
- Right lane nice: Drive in right lane and go off-road to avoid hitting other

For more details, checkout the videos posted on

http://www.cs.stanford.edu/~pabbeel/irl/

- Abbeel, P., & Ng, A. Y. (2004, July). Apprenticeship learning via inverse reinforcement learning. In Proceedings of the twenty-first International Conference on Machine Learning.

Your Moment of Zen


LOTS OF APPS LET YOU PLAN YOUR TRIPS USING REAL-TIME BUS, TRAIN, AND TRAFFIC DATA. THEY TRY TO PREDICT WHICH ROUTE WILL BE FASTER, BUT AREN'T ALWAYS RIGHT.
INSTEAD OF JUST *PLANNING*, MY NEW APP LETS YOU SEND "GHOST" VERSIONS OF YOU ALONG DIFFERENT ROUTES, SIMULATING THEIR TRAVEL USING THE REAL-TIME DATA
THAT WAY, YOU CAN SEE WHICH ROUTE TURNED OUT TO BE FASTER IN PRACTICE,
YOU CAN ALSO RACE YOUR PAST SELVES.
SOON...
UGH, LOST TO THE BIKE GHOST *AGAIN*.
HEY, MY KEY WON'T WORK.
I'M SORRY, BUT WE'VE DECIDED TO REPLACE YOU. THIS FLOATY GUY IS MUCH MORE PUNCTUAL.
BUT...
OUR *NEW* DAD NEVER MISSES OUR GAMES!
NOOO!