MDR Learning

Machine Learning
Deep Learning

Reinforcement Learning

Julien Simon
Global Evangelist, AI & Machine Learning, AWS
@julsimon

First things first

Artificial Intelligence: design software applications which exhibit human-like behavior, e.g. speech, natural language processing, reasoning or intuition

Machine Learning: using statistical algorithms, teach machines to learn from featurized data without being explicitly programmed

Deep Learning: using neural networks, teach machines to learn from complex data where features cannot be explicitly expressed

Types of Machine Learning

Supervised learning

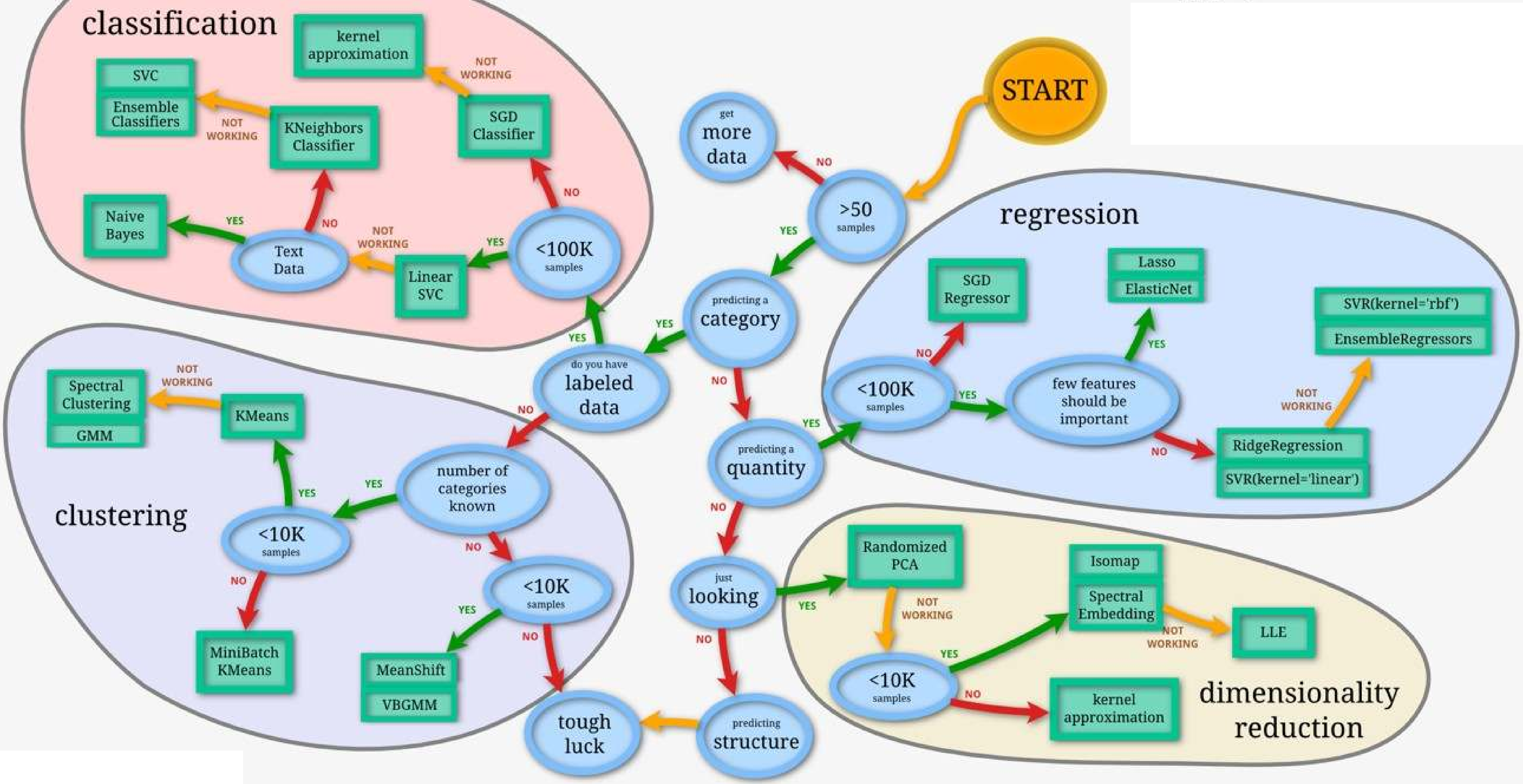
- Run an algorithm on a labeled data set.
- The model learns how to correctly predict the right answer.
- Regression and classification are examples of supervised learning.

Unsupervised learning

- Run an algorithm on an unlabeled data set.
- The model learns patterns and organizes samples accordingly.
- Clustering and topic modeling are examples of unsupervised learning.

Machine Learning in one slide

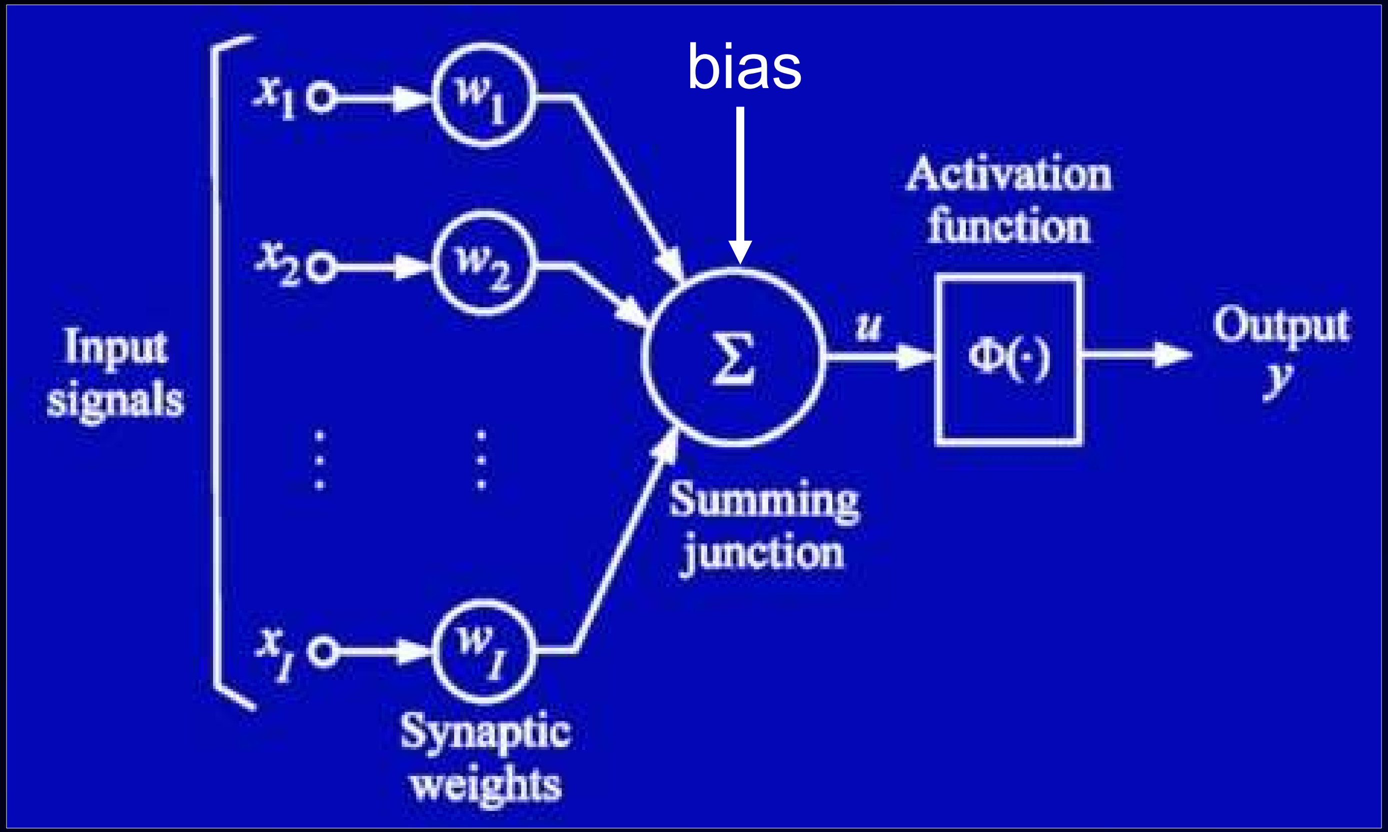
Presentation: https://fr.slideshare.net/JulienSIMON5/an-introduction-to-machine-learning-with-scikitlearn-october-2018 http://scikit-learn.org/stable/tutorial/machine_learning_map/



Deep Learning in five slides

Presentation: https://fr.slideshare.net/JulienSIMON5/an-introduction-to-deep-learning-84214689

The neuron



$$\sum_{i=1}^{I} x_i * wi + b = u$$

"Multiply and Accumulate"

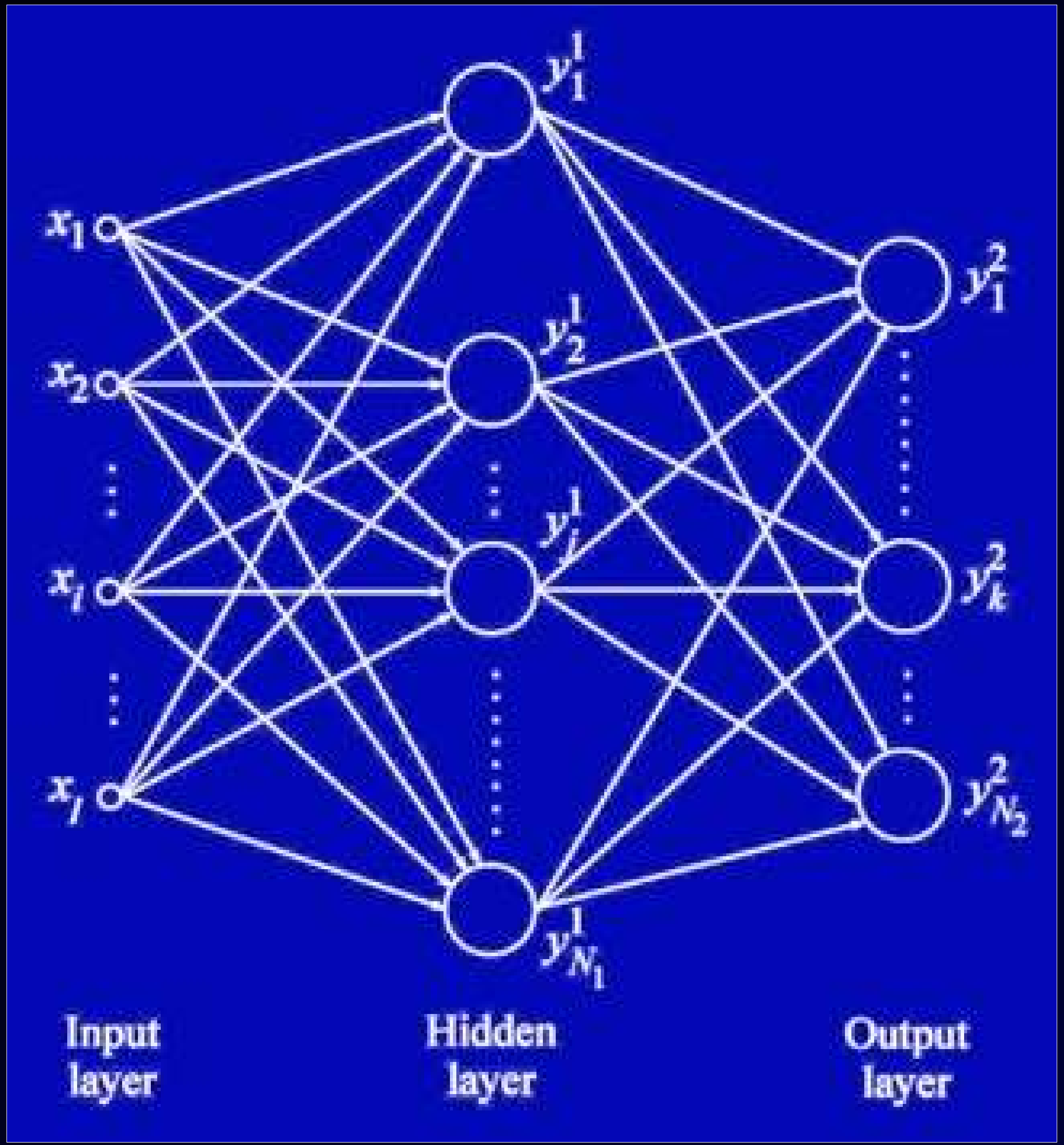
Activation functions

Name	Plot	Equation
Identity		$f(x) = x$
Binary step		$f(x) = \begin{cases} 0 & \text{for} & x < 0 \\ 1 & \text{for} & x \geq 0 \end{cases}$
Logistic (a.k.a. Soft step)		$f(x) = \frac{1}{1+e^{-x}}$
TanH		$f(x) = \tanh(x) = \frac{2}{1+e^{-2x}} - 1$
ArcTan		$f(x) = \tan^{-1}(x)$
Softsign [7][8]		$f(x) = \frac{x}{1+\lvert x \rvert}$
Rectified linear unit (ReLU)[9]		$f(x) = \begin{cases} 0 & \text{for} & x < 0 \\ x & \text{for} & x \geq 0 \end{cases}$

Source: Wikipedia

Neural networks

Building a simple classifier



l features

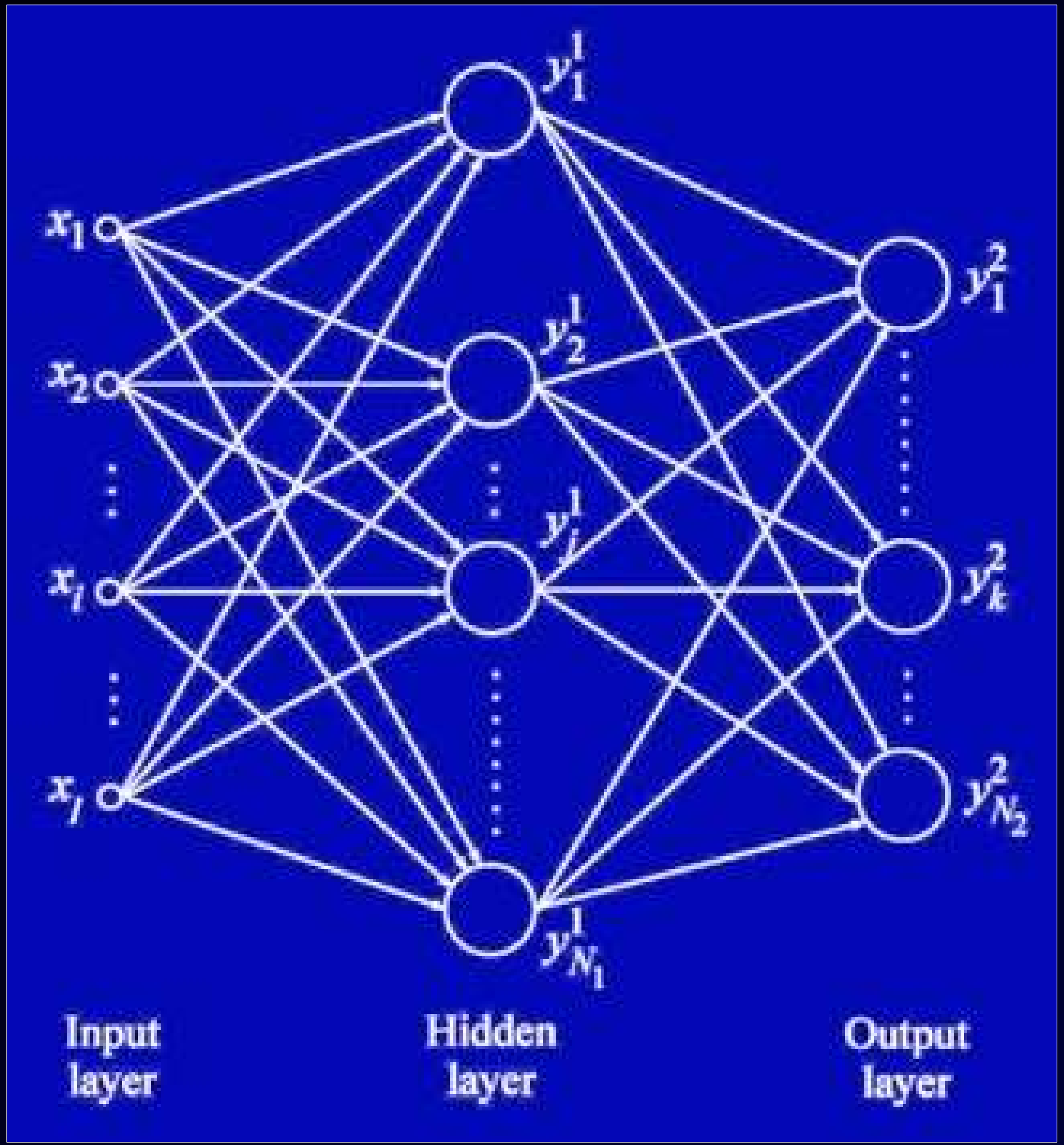
$$
X = \left.\begin{bmatrix} X_{11}, X_{12}, \dots X_{1l} \\ X_{21}, X_{22}, \dots X_{2l} \\ \dots\ \dots\ \dots \\ X_{m1}, X_{m2}, \dots X_{ml} \end{bmatrix}\right\} \text{m samples}
$$

$$
y = \begin{bmatrix} 2 \\ 0 \\ \dots \\ 4 \end{bmatrix} \quad \left.\begin{bmatrix} 0,0,1,0,0,\dots,0 \\ 1,0,0,0,0,\dots,0 \\ \dots \\ 0,0,0,0,1,\dots,0 \end{bmatrix}\right\} \text{m labels, } N_2 \text{ categories}
$$

One-hot encoding

Neural networks

Building a simple classifier



l features

$$X = \begin{bmatrix} x_{11}, & x \ldots & \ldots x_{1l} \\ x_{21}, & x \ldots & \ldots x_{2l} \\ \ldots & \ldots & \\ x_{m1}, & x \ldots & \ldots \end{bmatrix}$$

m samples

$x_{11}, x_{12}, \ldots, x_{1l}$

$$y = \begin{bmatrix} 2 \\ 0 \\ \ldots \\ 4 \end{bmatrix} \quad \begin{bmatrix} 0,0,1,\ldots, 0 \\ 1,0,0,\ldots, 0 \\ \ldots \\ 0,0,0,\ldots, 0 \end{bmatrix}$$

m labels, N_2 categories

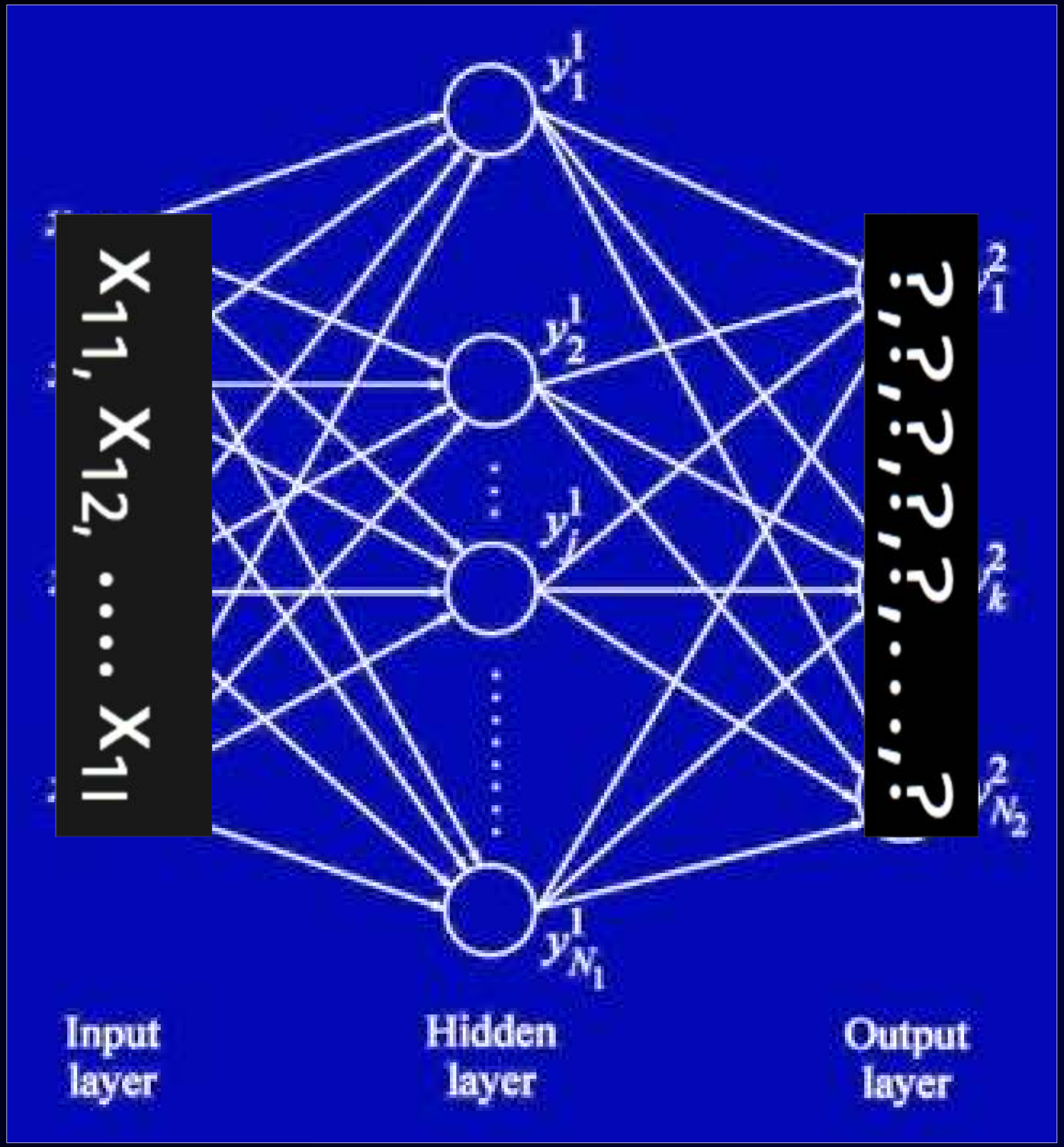
0,0,1,0,0,…,0

One-hot encoding

$$\text{Accuracy} = \frac{\text{Number of correct predictions}}{\text{Total number of predictions}}$$

Neural networks

Building a simple classifier



Weights are initialized at random

→ the initial network predicts at random

$f(X_1) = Y'_1$

A loss function measures the difference between the real label Y_1 and the predicted label Y'_1

error = $\text{loss}(Y_1, Y'_1)$

For a batch of samples:

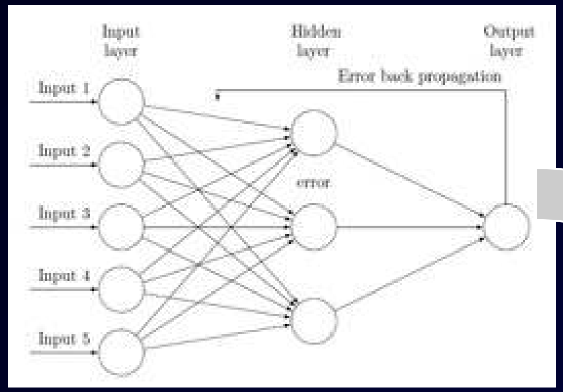
$$\sum_{i=1}^{\text{batch size}} \text{loss}(Y_i, Y'_i) = \textbf{batch error}$$

The purpose of the training process is to minimize error by gradually adjusting weights.

Training a neural network

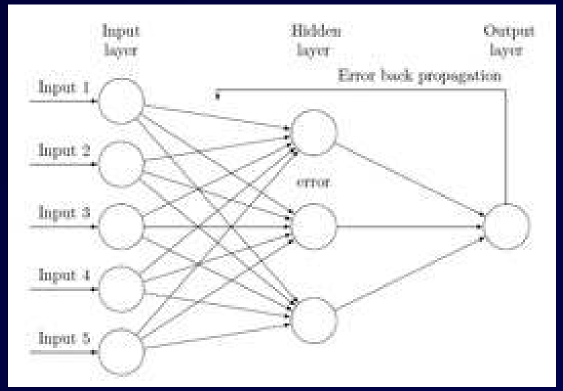
Training data set

Forward propagation



Backpropagatio
n

Training



Trained
neural network

Batch size
Learning rate
Number of epochs

Hyper parameters

Why Reinforcement Learning?

Building a dataset is not always an option

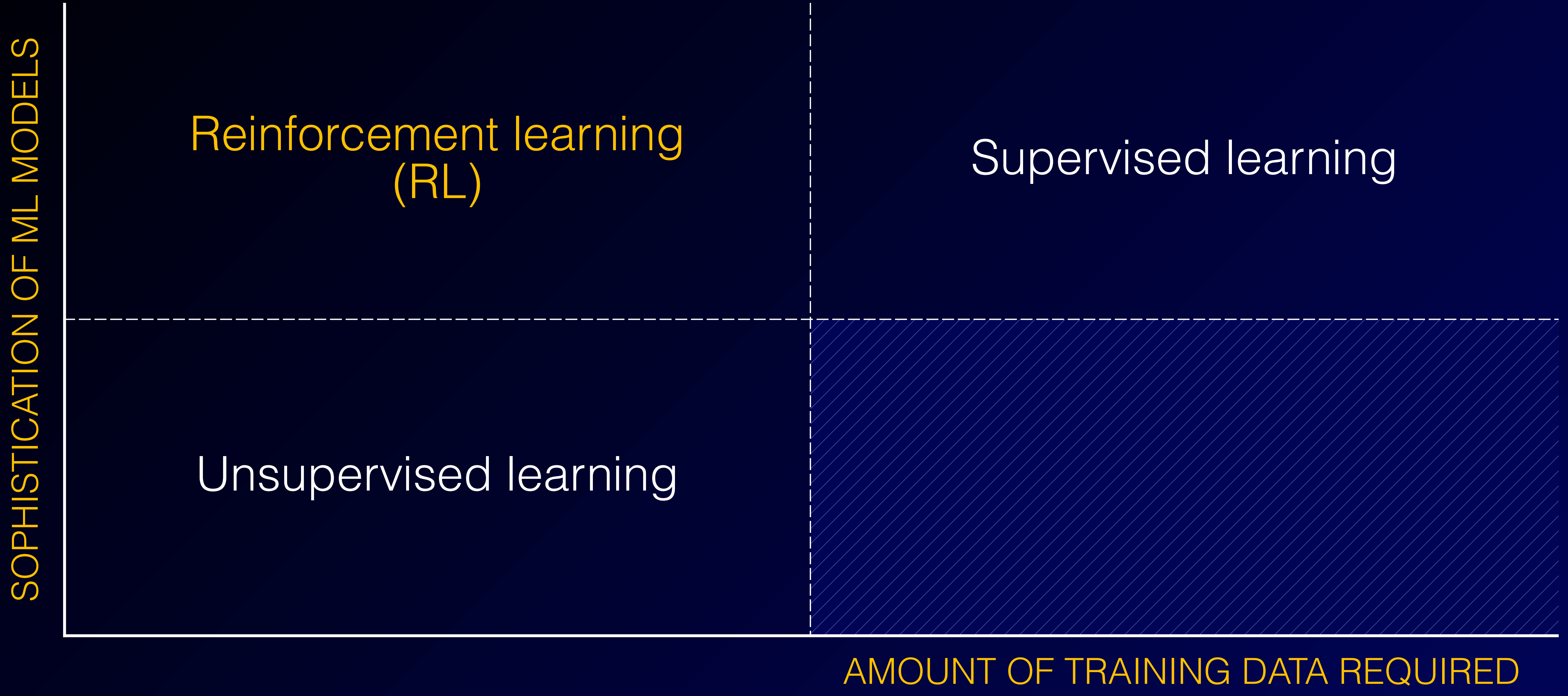
Large, complex problems

Uncertain, chaotic environments

Continuous learning

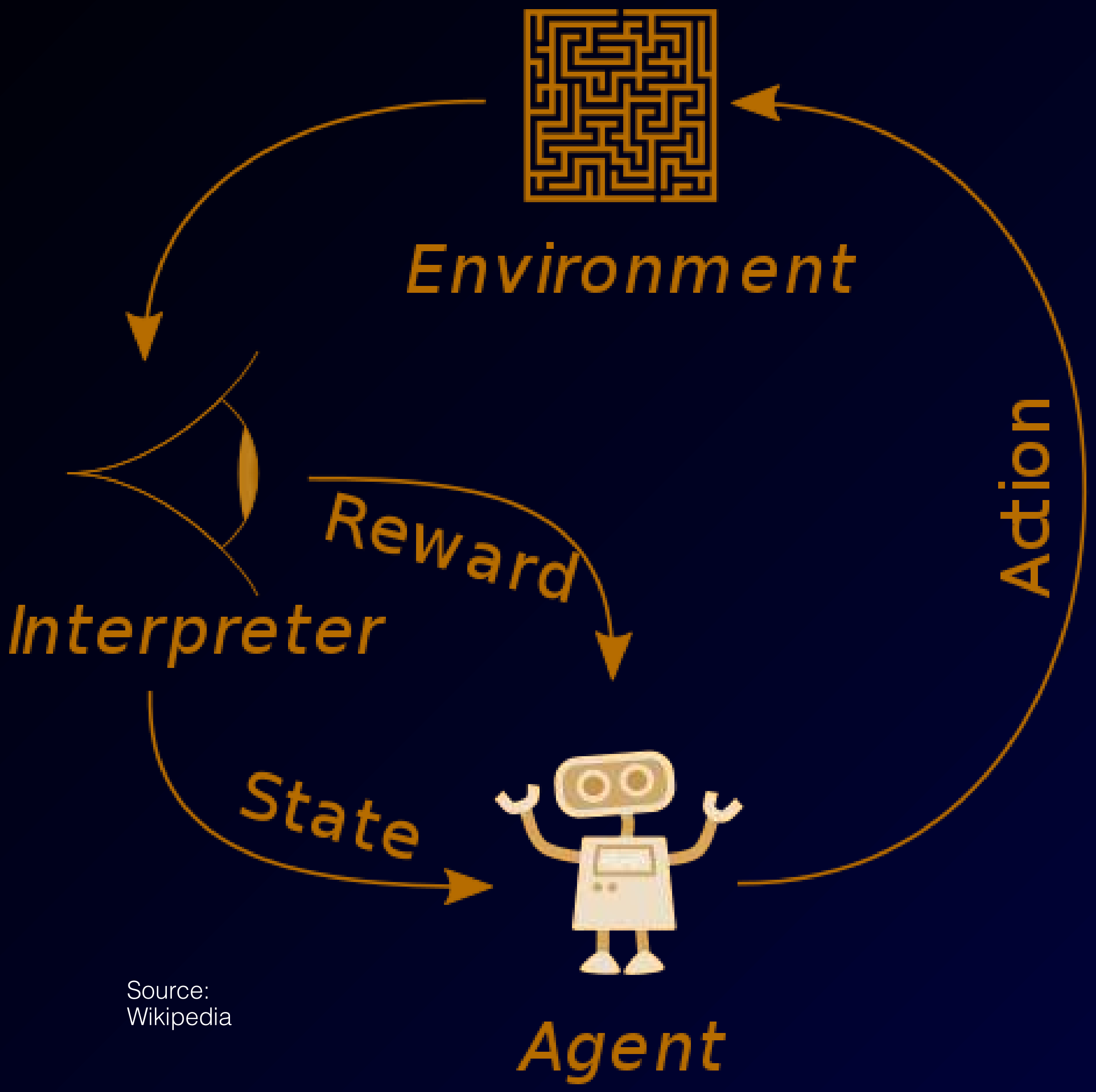
Supply chain management, HVAC systems, industrial robotics, autonomous vehicles, portfolio management, oil exploration, etc.

Types of Machine Learning



Learning without any data: we've all done it!

Reinforcement Learning



Source: Wikipedia

An agent interacts with its environment.

The agent receives positive or negative rewards for its actions: rewards are computed by a user-defined function which outputs a numeric representation of the actions that should be incentivized.

By trying to maximize the accumulation of rewards, the agent learns an optimal strategy (aka policy) for decision making.

Training a RL model

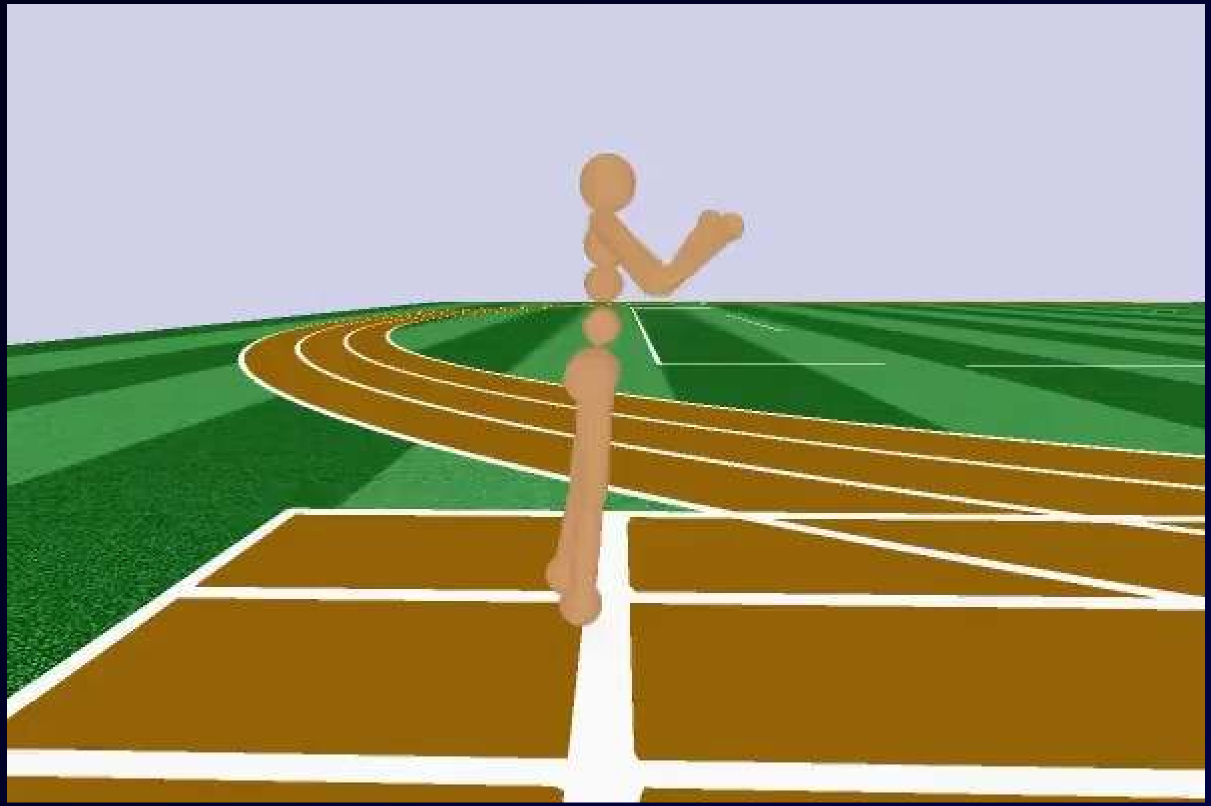
1. Formulate the problem: goal, environment, state, actions, reward
2. Define the environment: real-world or simulator?
3. Define the presets
4. Write the training code and the reward function
5. Train the model

RL example: learning to walk

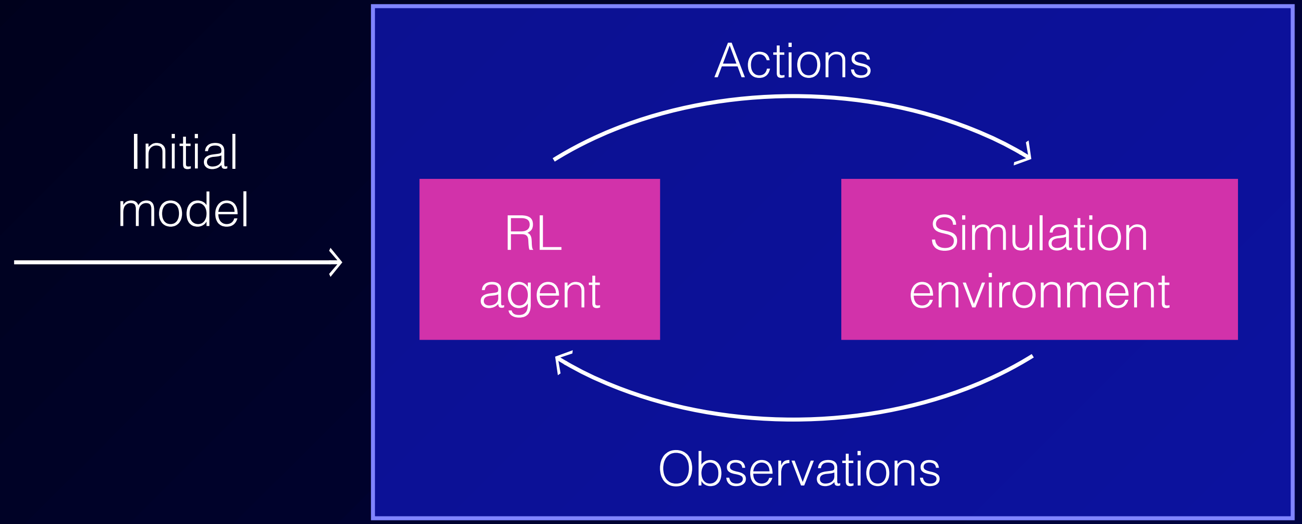
https://github.com/awslabs/amazon-sagemaker-examples/tree/master/reinforcement_learning/rl_roboschool_ray

The players

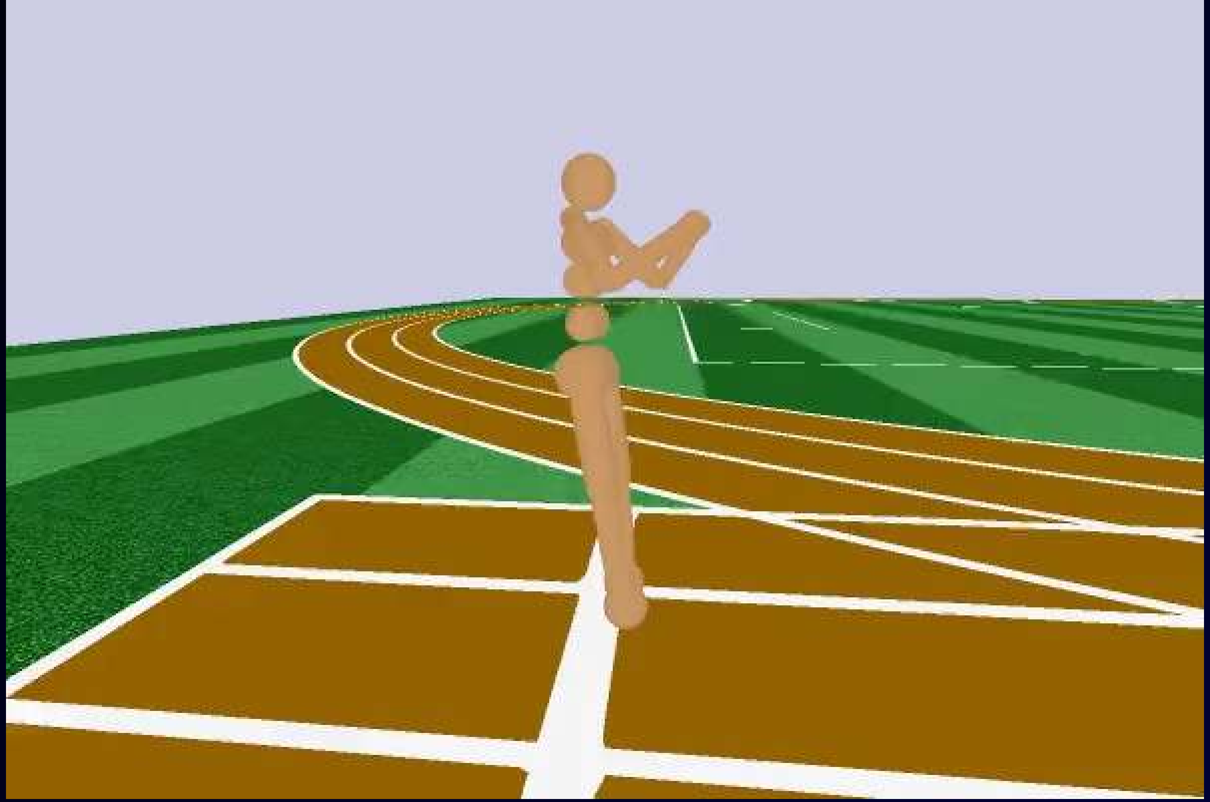
At first, the agent can't even stand up



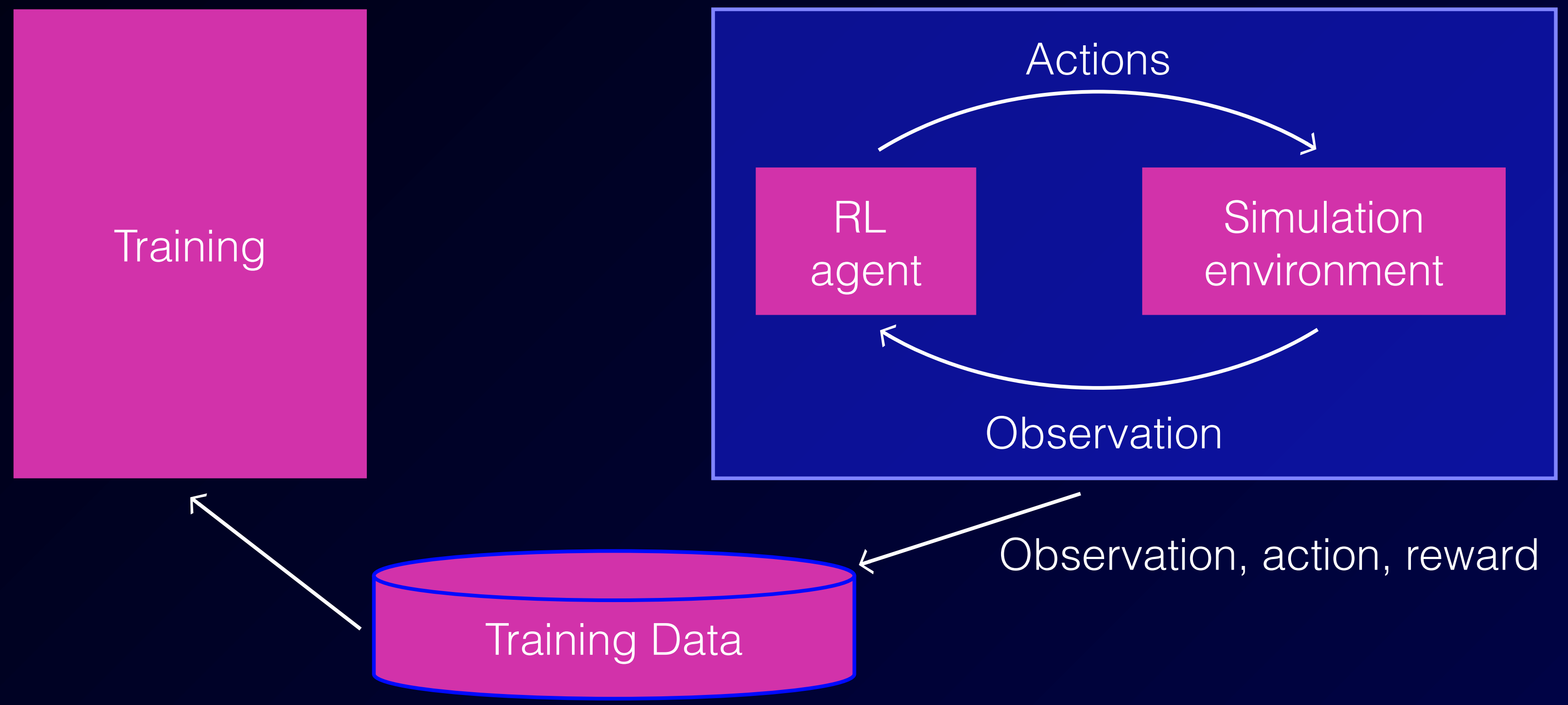
Actions and observations



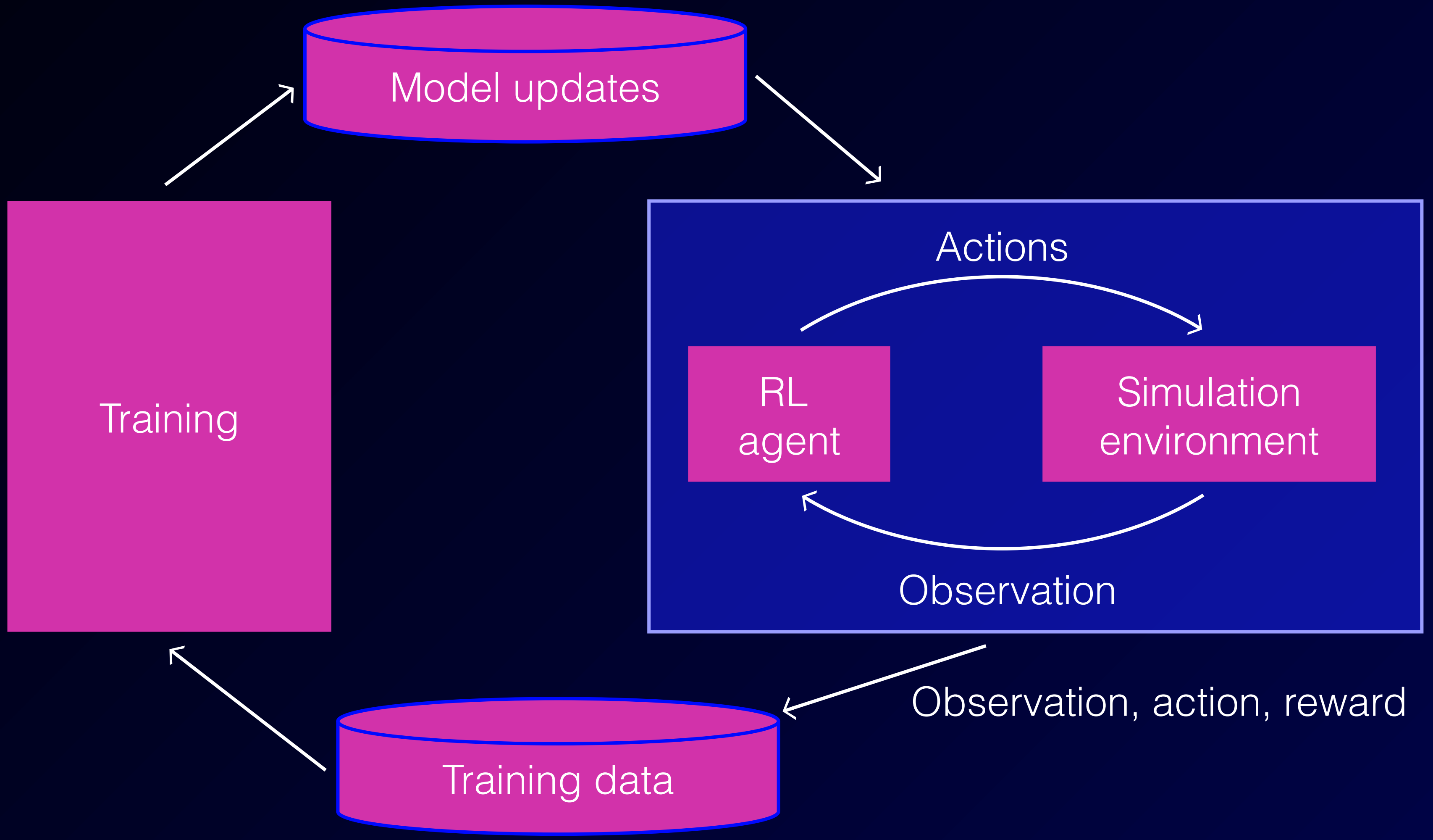
The model learns through actions and observations



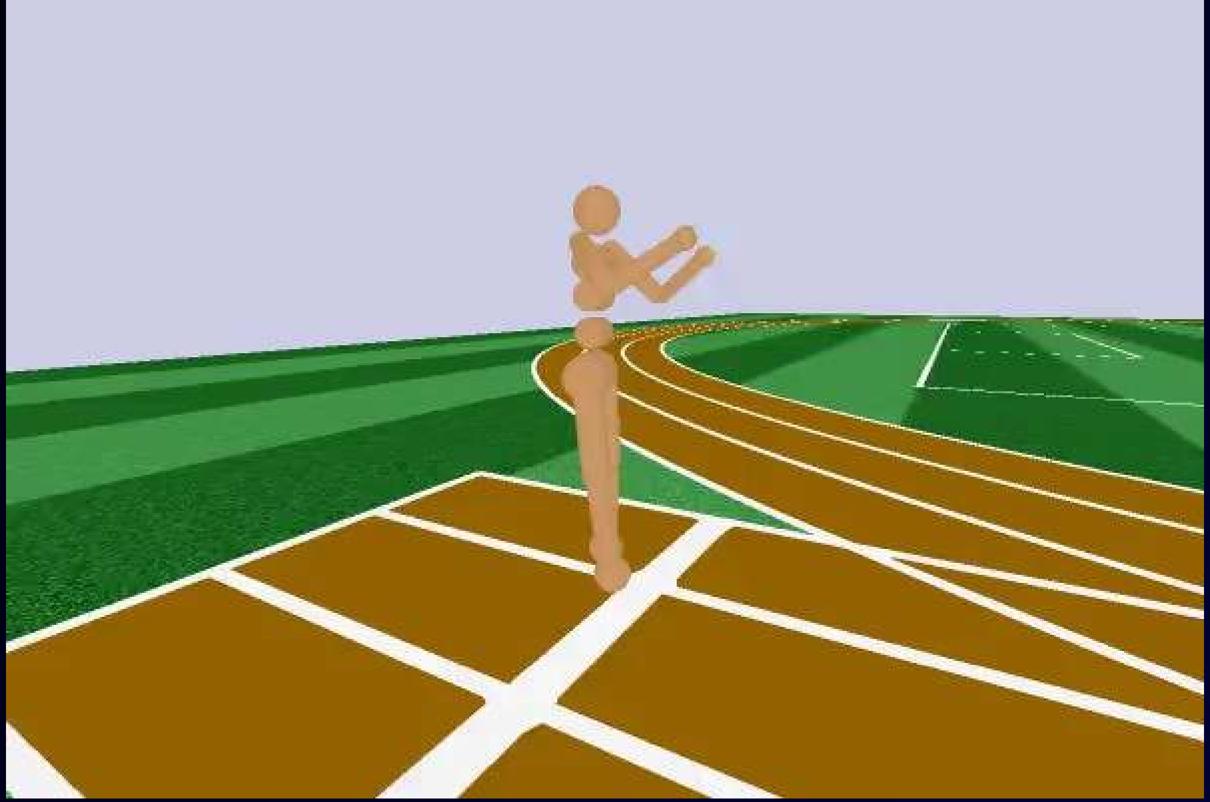
Interactions generate training data

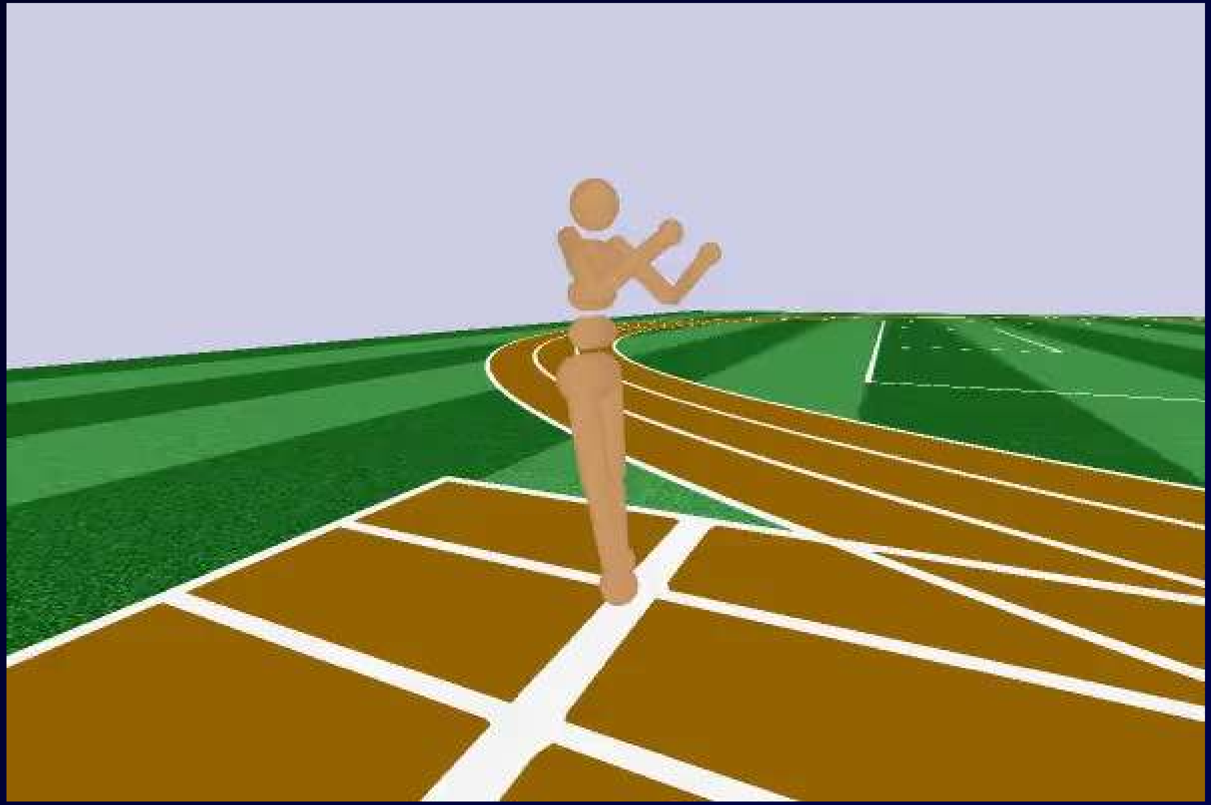


Training results in model updates

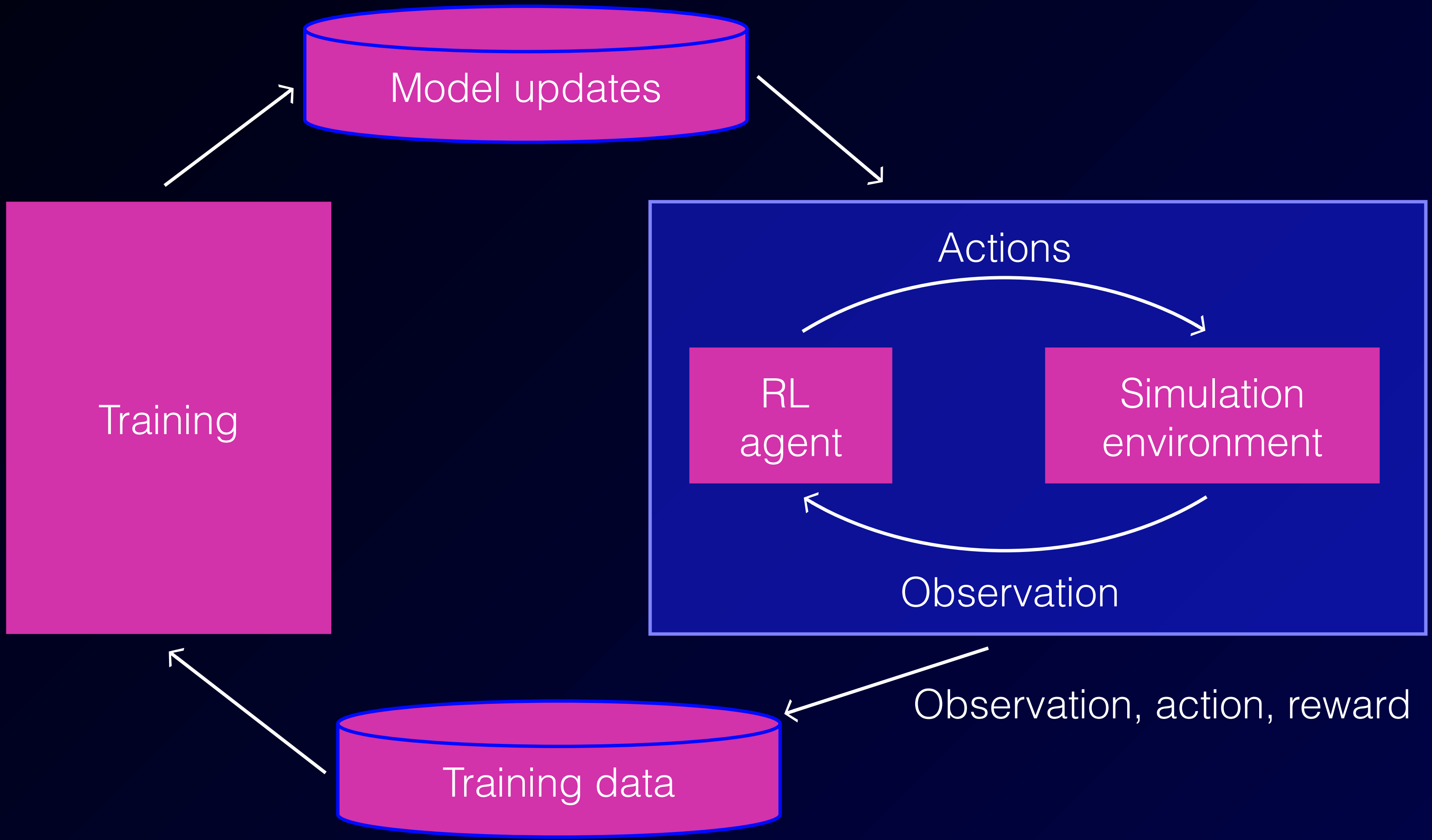


The agent learns to stand and step

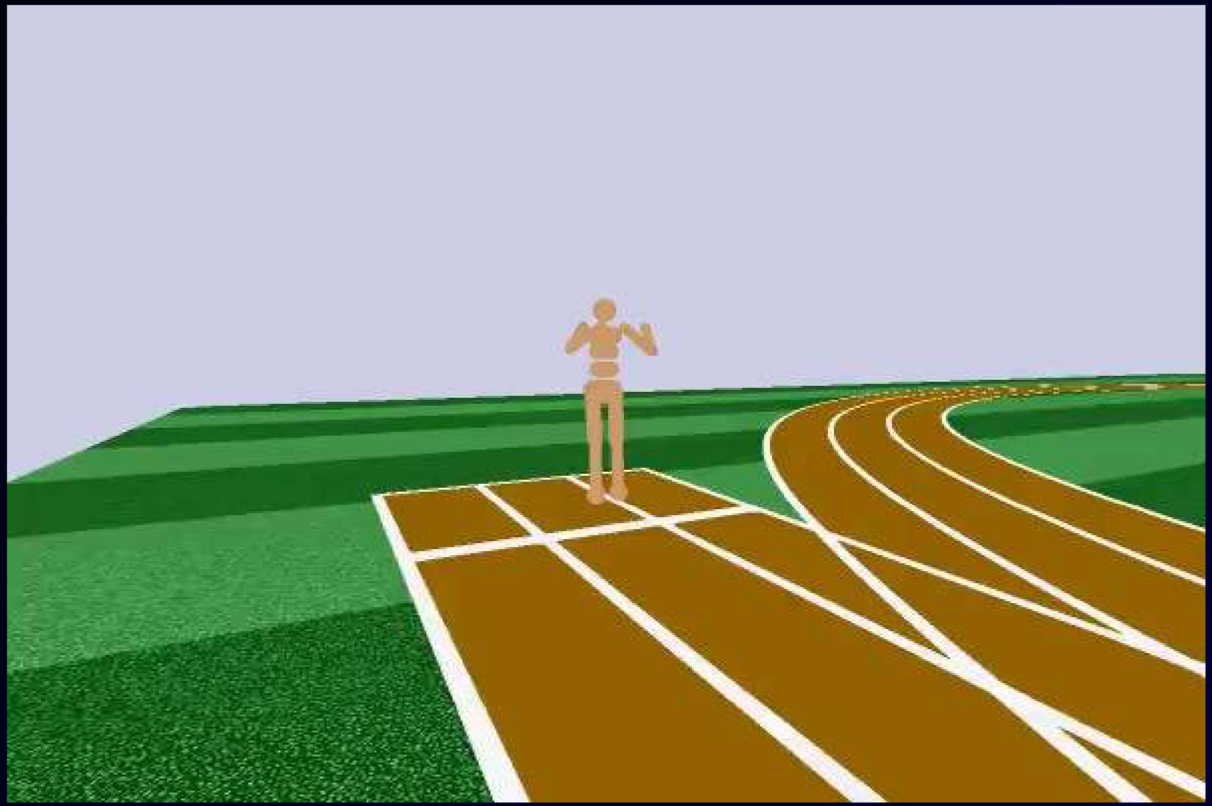


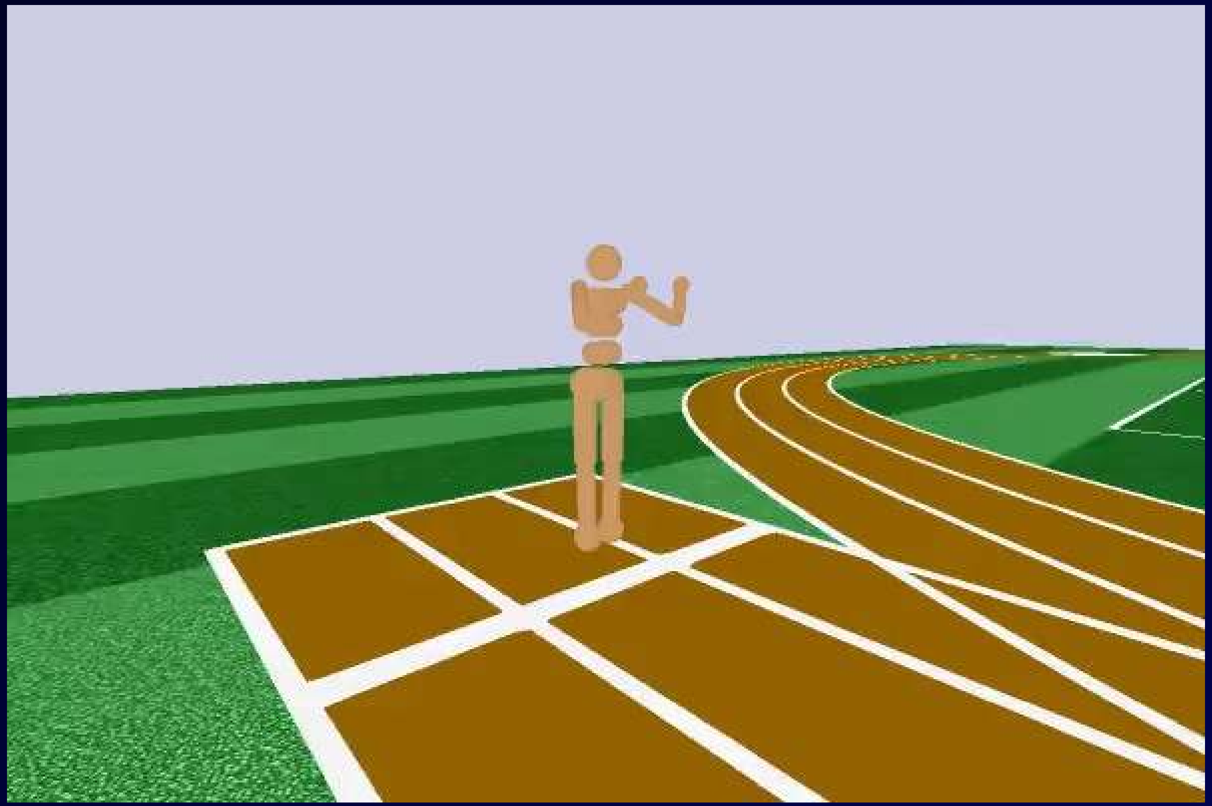


Multiple training episodes improve learning

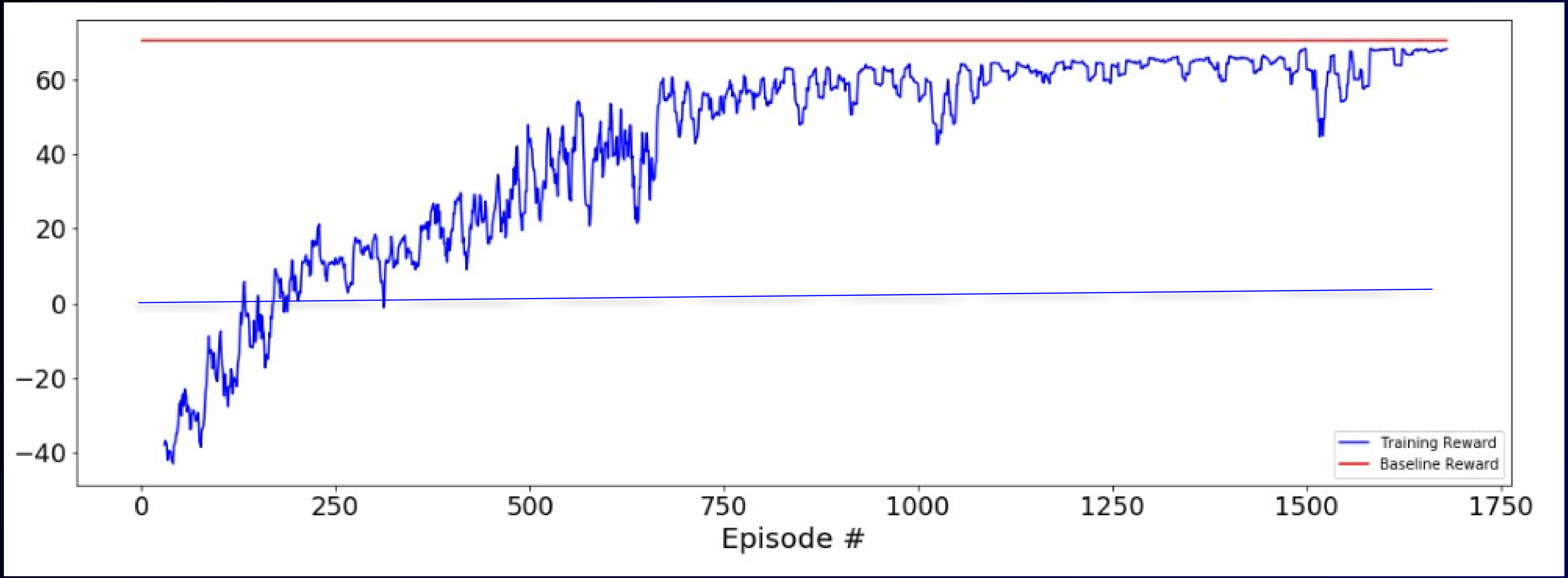


Making progress

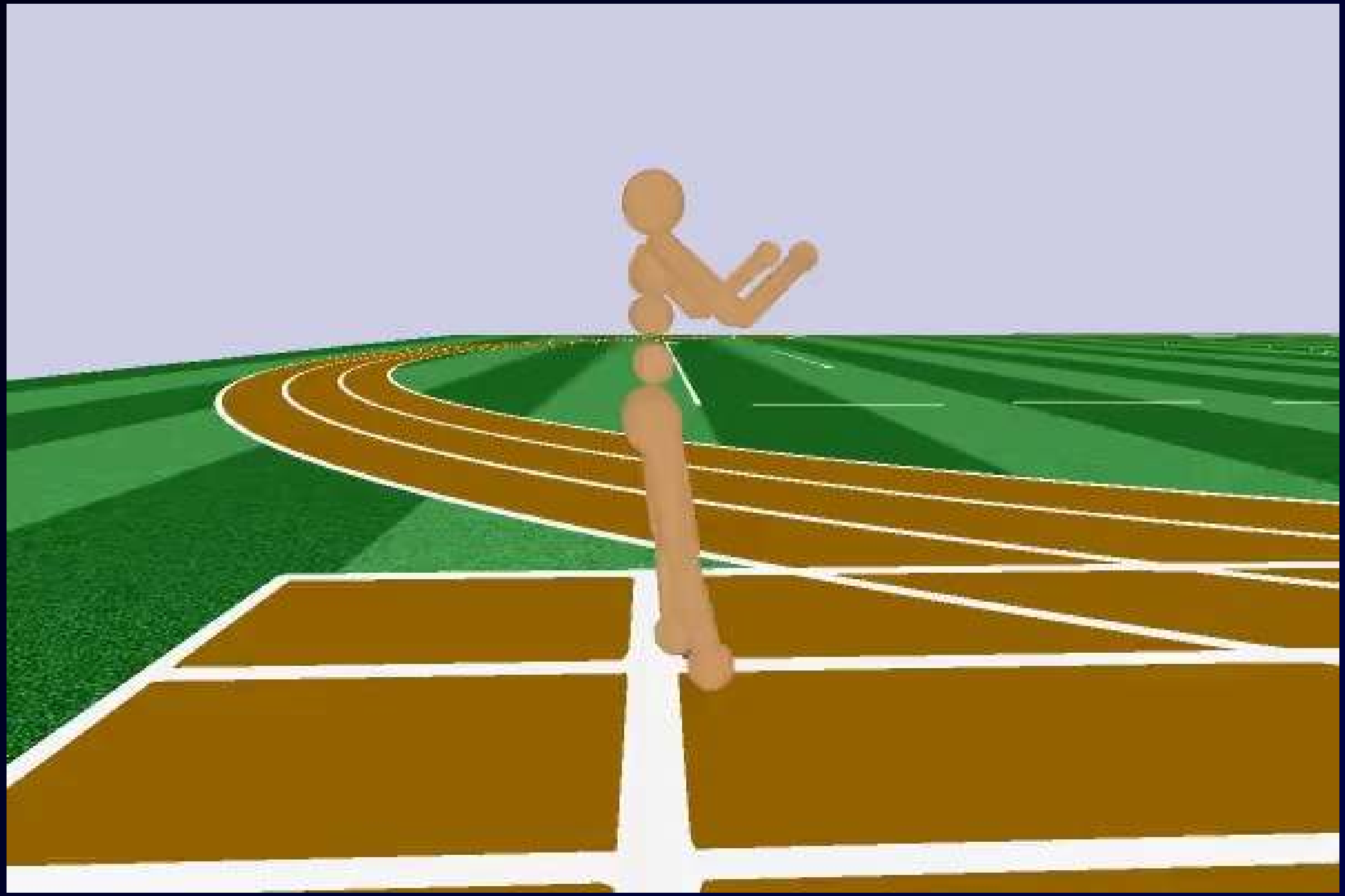




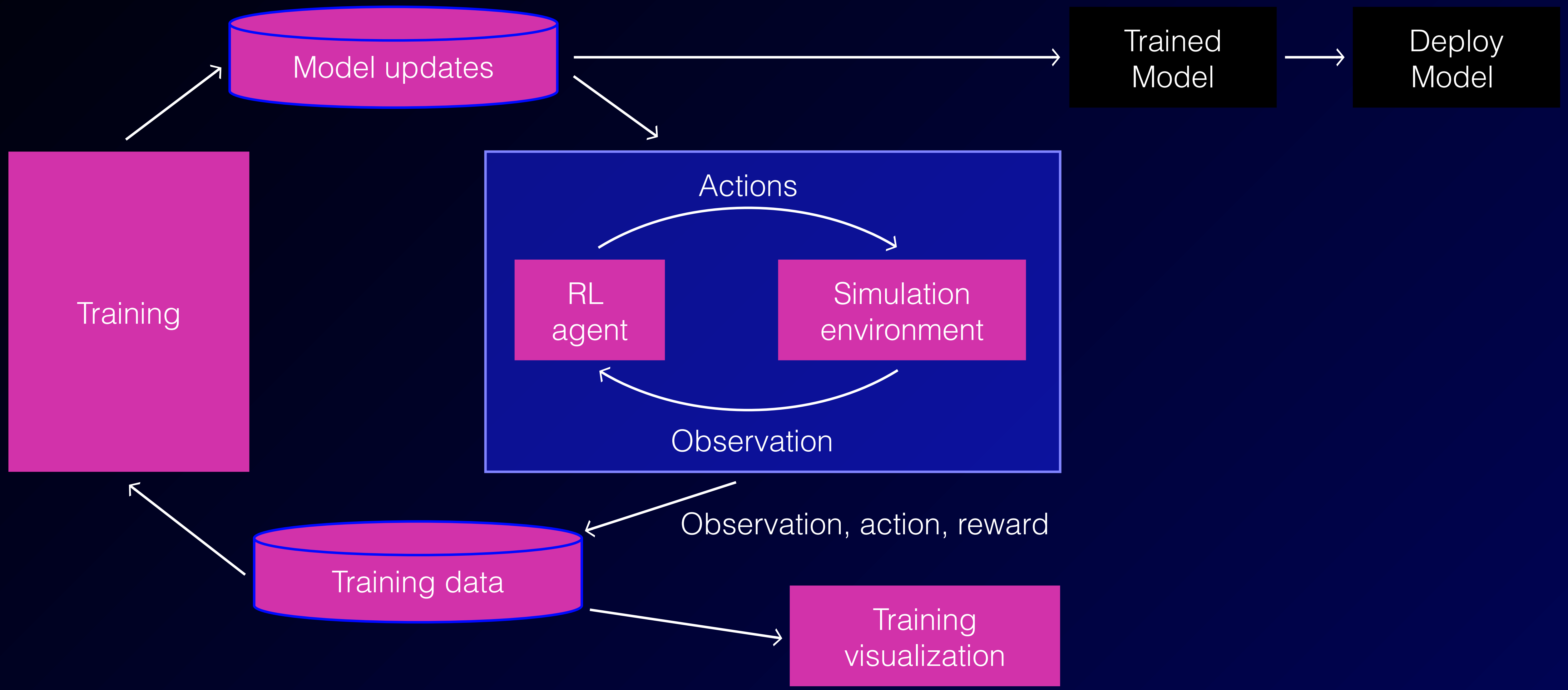
RL agents try to maximize rewards



Eventually, the model learns how to walk and run



Evaluate and deploy trained models



Real-life use cases for RL

Robotics

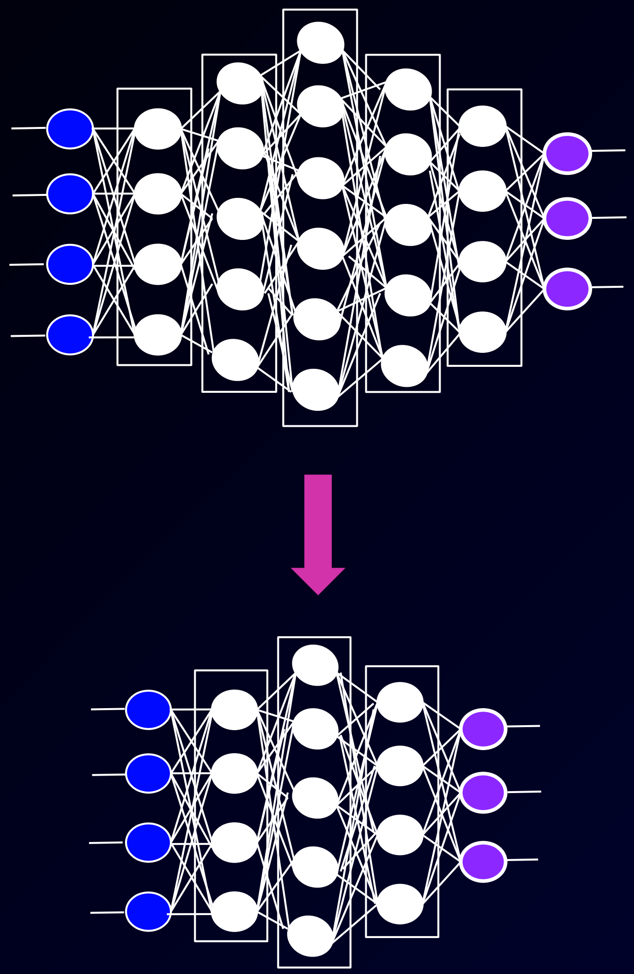
Financial portfolio management

Objective	**Maximize the value of a financial portfolio**
STATE	Current stock portfolio, price history
ACTION	Buy, sell stocks
REWARD	Positive when return is positive Negative when return is negative

Jiang, Zhengyao, Dixing Xu, and Jinjun Liang « A deep reinforcement learning framework for the financial portfolio management problem. » arXiv:1706.10059 (2017)

https://github.com/awslabs/amazon-sagemaker-examples/tree/master/reinforcement_learning/rl_portfolio_management_coach_customEnv

Compressing deep learning models

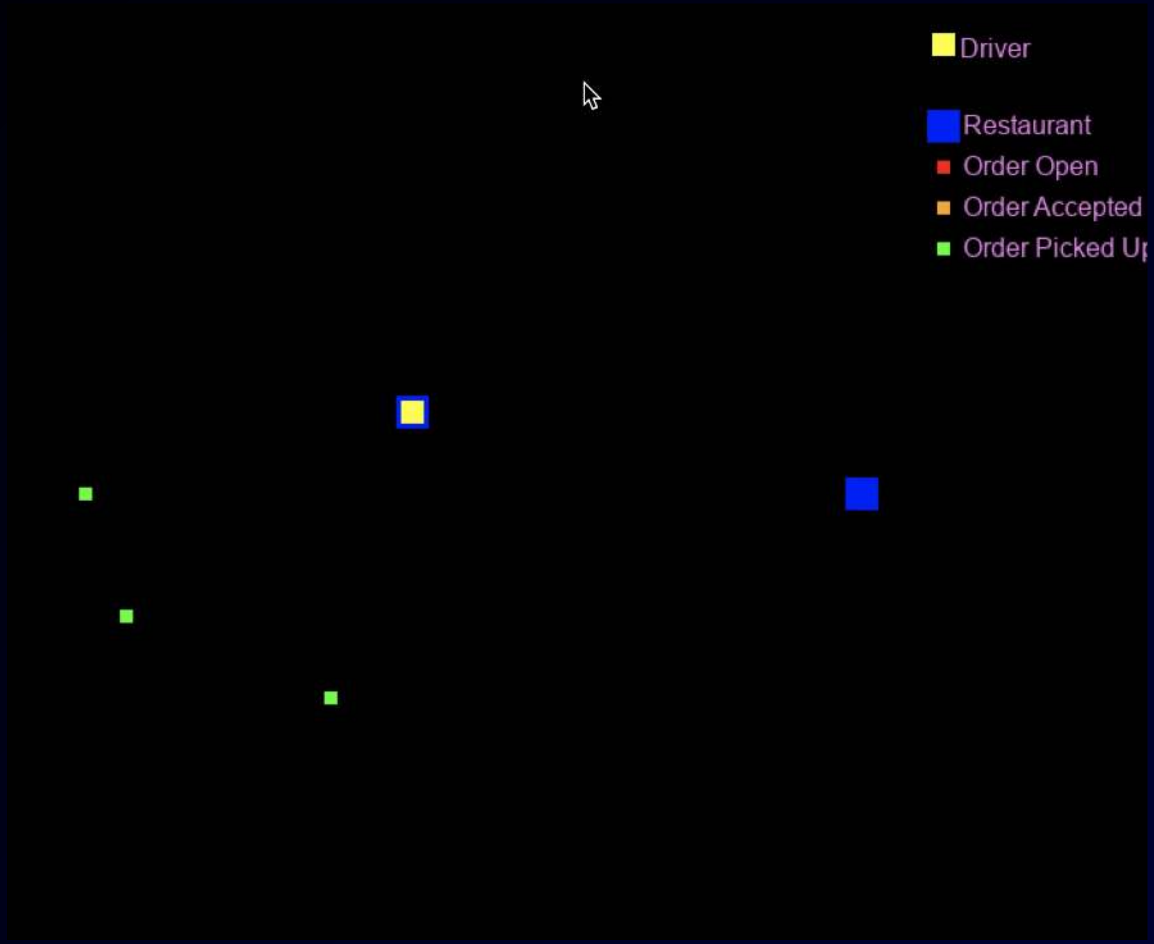


Objective	**Compress model without losing accuracy**
STATE	Layers
ACTION	Remove or shrink a layer
REWARD	A combination of compression ratio and accuracy.

Ashok, Anubhav, Nicholas Rhinehart, Fares Beainy, and Kris M. Kitani "N2N learning: network to network compression via policy gradient reinforcement learning." arXiv:1709.06030 (2017).

https://github.com/awslabs/amazon-sagemaker-examples/tree/master/reinforcement_learning/rl_network_compression_ray_custom

Vehicle routing



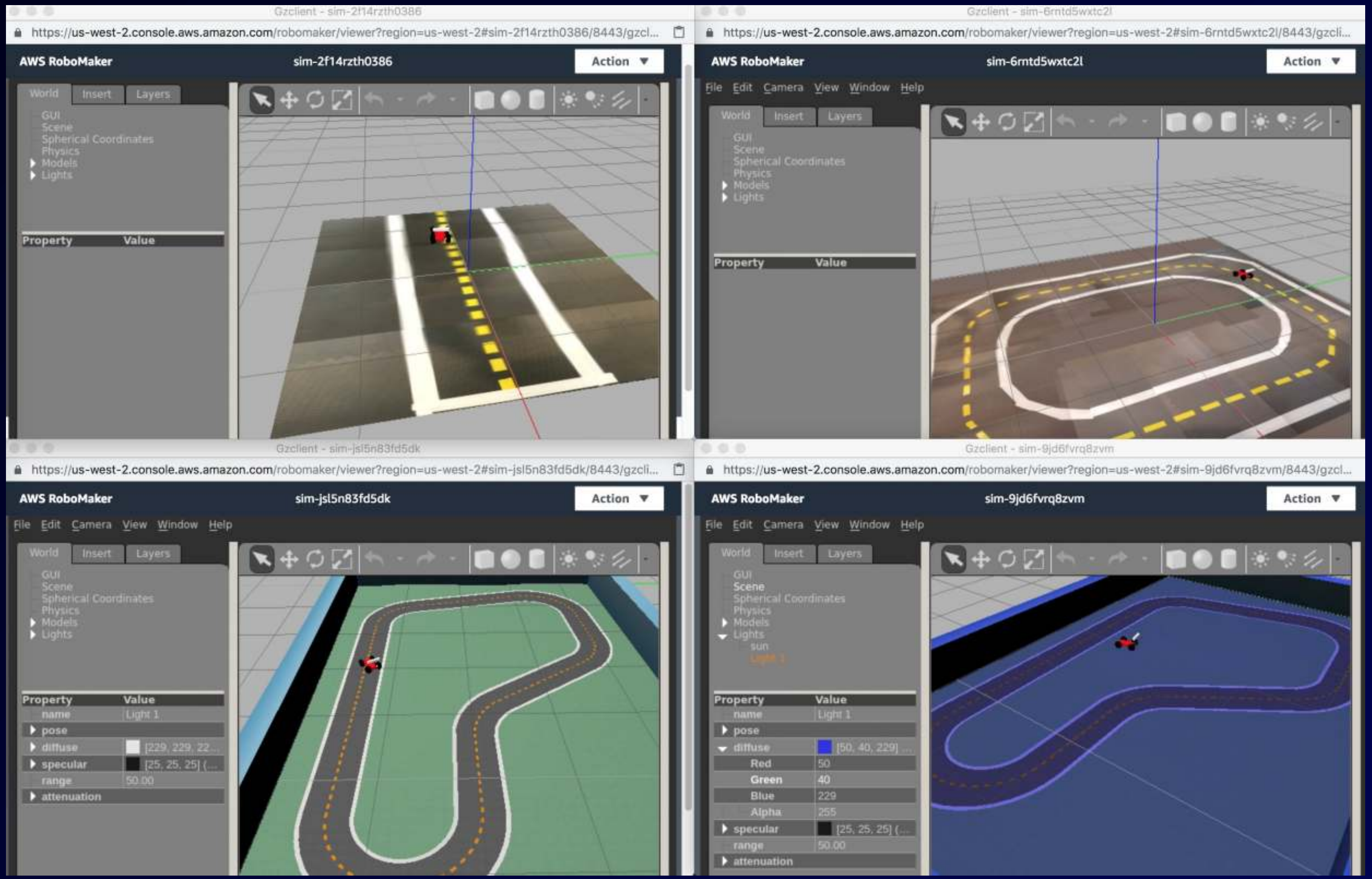
Objective	**Fulfill customer orders**
STATE	Current location, distance from homes
ACTION	Accept, pick up, and deliver order
REWARD	Positive when we deliver on time Negative when we fail to deliver on time

https://github.com/awslabs/amazon-sagemaker-examples/tree/master/reinforcement_learning/rl_traveling_salesman_vehicle_routing_coach

Autonomous driving

AWS DeepRacer

1/18th scale autonomous vehicle



Amazon RoboMaker

Getting started

http://aws.amazon.com/free

https://ml.aws

https://aws.amazon.com/sagemaker
https://github.com/awslabs/amazon-sagemaker-examples
https://aws.amazon.com/blogs/aws/amazon-sagemaker-rl-managed-reinforcement-learning-with-amazon-sagemaker/

https://aws.amazon.com/deepracer/

https://medium.com/@julsimon

https://amzn.to/2mp1Lf5

Thank you!

Julien Simon
Global Evangelist, AI & Machine Learning, AWS
@julsimon