What Is Reinforcement Learning?

Supervised Learning: Puts a name to the face (list of features -> label). “Based on the color and shape of this thing, it must be a cheeseburger”.

Un-Supervised Learning: Can identify things based on correlations and patterns of previous examples. “This thing is like that other thing”.

Reinforcement Learning: Actions based on short- and long-term rewards. “I should eat this cheeseburger because it tastes good and will help me live”.

Reinforcement Learning is different from Supervised and Un-Supervised Learning because it continually trains itself using **trial and error** to reach a **reward—**it has the unique opportunity to explore its **environment**. The result of a converged Supervised/Un-Supervised Learning algorithm is called a *model*, whereas the result of a Reinforcement Learning algorithm is called an **agent**. The goal of a Reinforcement Learning **agent** is to correlate immediate actions with the returns they will reap down the road—to pick the best possible **action** for the given current **state**.

Reinforcement Learning agents are ideal for learning to play video games like Mario:

- Mario’s possible **actions** are walk left/right, run left/right, and jump
- The **reward** in the game is the points. Mario’s goal is to maximize his total number of points. This can be done by collecting coins, reaching the castle quickly, and stomping on goombas.
- There is no relevant word for it in Reinforcement learning, but we will call the opposite of reward a **punishment**. Mario lives in a high stakes world so his **punishment** is dying. This can happen when he falls out of bounds, hits a goomba, or is incinerated by fire balls.
- All rewards collected by Mario will be classified as a **positive action** and all punishments Mario encounters will be classified as a **negative action**.
- Mario’s **environment** will include his location, enemy locations, coin locations, world boundaries, etc. These will all be included in the **agent’s inputs** which is its current **state**.

At its core, Reinforcement Learning is a Convolutional Neural Network whose goal is to predict which action will reap the best rewards. To train Mario, the Reinforcement algorithm would start by applying random actions to Mario in order to observe and slowly learn how his **actions** correlate with his **rewards** and **punishments**. These **actions** will slowly become less random as the Neural Network gets better at understanding Mario’s **environment**.

Relevant Literature

Socially Aware Motion Planning with Deep Reinforcement Learning Video Link | Journal Link
This is an ambitious self-driving robot developed by MIT, taught using Reinforcement Learning. It drives towards a destination, but with a twist. Its actions not only involving driving towards he destination, but also to follow social norms. These social norms include keeping from getting too close to a human's personal space and biasing its driving to the right side of the hall (left on left driving countries). Rewards are following the projected path, penalties are not following social norms. They also noted that the coordinate system the robot used involved having the x-axis be the direction that the destination was located.

A Survey of Machine Learning Approaches to Robotic Path-Planning Journal Link
This is the UC Boulder journal you sent me. What I think is interesting about their Reinforcement Learning approach is that the agent's only potential actions are to follow the potential path, avoid obstacle, and recover from collision. They do not use Machine Learning to *create* the path, but they do use Machine Learning to determine a strategy to *follow* the path.

Self-supervised Deep Reinforcement Learning for Robot Navigation Video Link
Uses Reinforcement Learning just to explore a loop of a hallway. The reward is continuing counterclockwise around a hallway, its punishment is colliding with something. Nothing too special about this one, just clean and simple.

Autonomous Drifting using Machine Learning Video Link
MIT used Reinforcement Learning paired with computer vision to create a RC car that can drift in a desired radius. Since Reinforcement Learning is dependent on immediate action based on its current state, the car had no problem drifting between regular floor and a portion of floor covered in slippery rock salt. It can also easily rebound from light collisions with obstacles. The agent is taught using a combination of a complex virtual simulation and real-world examples. This just increased the real-world learning efficiency. Its reward is reducing the "cost" of its drift. This cost was not defined, but I would guess that it is how close it is to its desired path (perfect path has a cost of 0, terrible path has a high cost).