Machine Learning Applications for Demand Forecasting and Inventory Replenishment

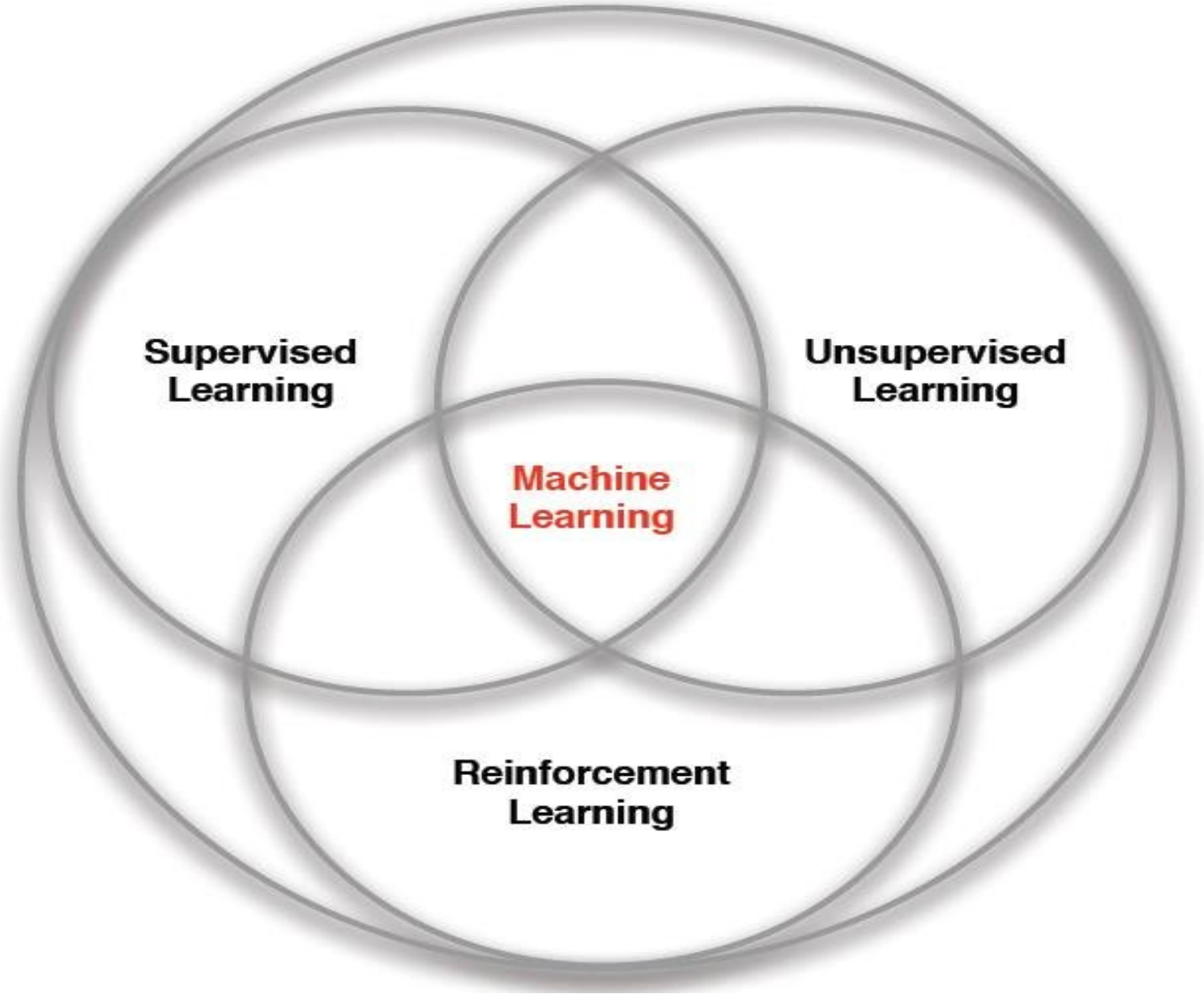
Ashwin Rao

Chief Science Officer @ Wayfair and Adjunct Professor @ Stanford University

March 20, 2022

1. Machine Learning Overview
2. Supply-Chain Headaches
3. Demand Forecasting: Transformers and Embeddings to the rescue
4. Inventory Replenishment: Reinforcement Learning to the rescue

Machine Learning



Machine Learning Overview

- Supervised Learning: Predictions based on learning data implications
- Unsupervised Learning: Unearthing patterns & structures within data
- Reinforcement Learning: Optimal Decisioning under Uncertainty
- ML has been a breakthrough for images, text, games
- Largely due to the recent success of Deep Neural Networks
- We are now adapting Deep Learning techniques to other domains
- Works well when we have plenty of data and plenty of compute
- ML practice tends to be quite laborious on Big Data Engineering
- I'm seeing promising results in Finance and Retail applications
- I cover this in depth in my new book

Supply-Chain Headaches and Machine Learning Pills

- Customer Demand is hard to forecast
- Lately, Supply constraints pose challenges too
- Often, Space and Throughput constraints
- Network Planning, Labor Planning, Transportation - all challenging
- Tracking Inventory Count/Location is hard
- Not easy to control damage, theft, misplacement, spoilage
- Nirvana: Inventory at right time/place, in right quantity, at low cost
- Blend of Traditional OR + Modern ML paves the path to Nirvana
- Today we look at two game-changing applications of ML
- Demand Forecasting and Inventory Replenishment

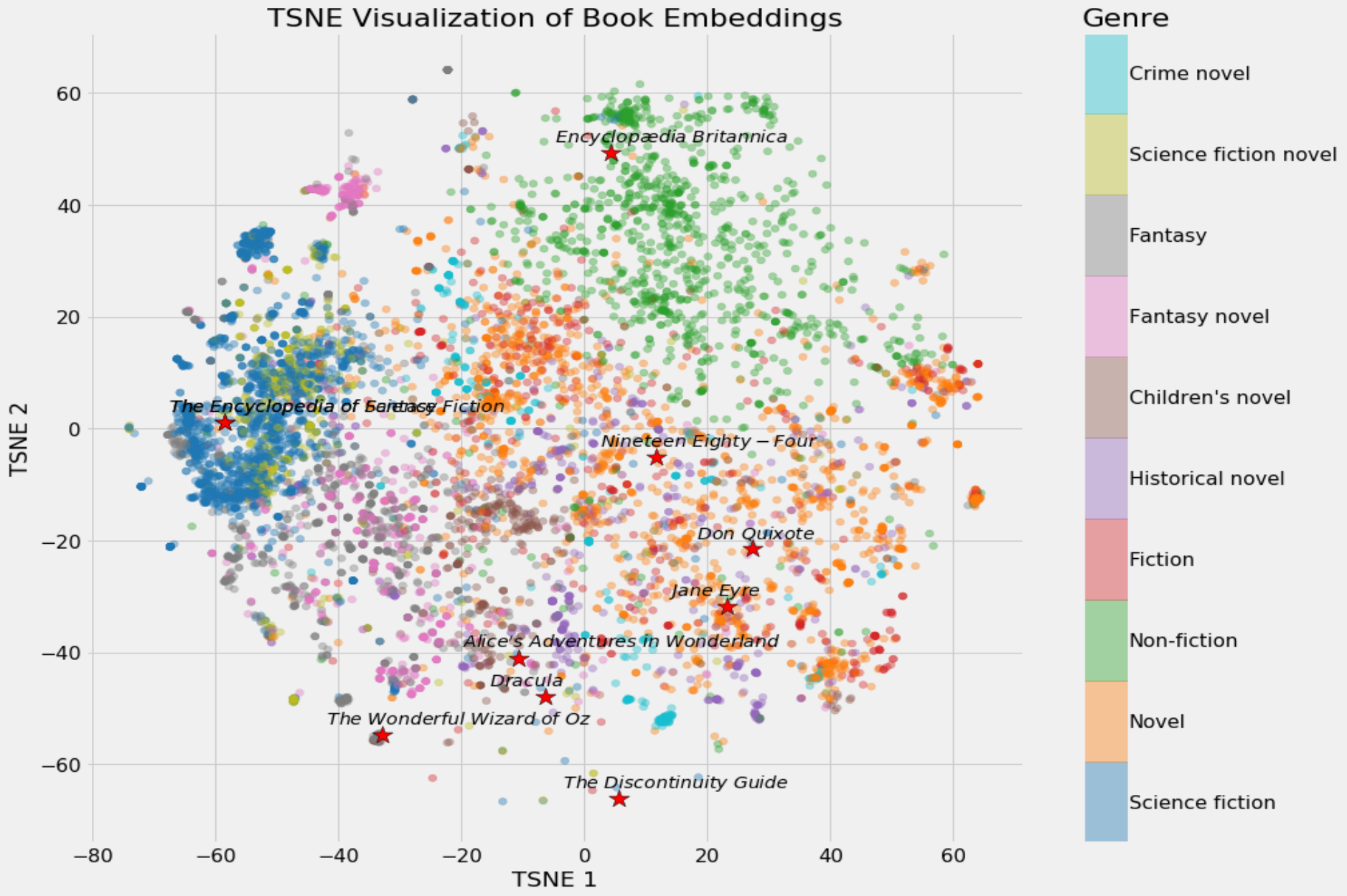
Demand Forecasting

- Why do we care about Demand Forecasting?
- Because it is foundational to LOTs of decisions in Supply-Chain
- Design/Planning Decisions and Operational/Control Decisions
- Design/Planning examples: Network, Labor, Transportation
- Operational/Control examples: Replenishment, Logistics
- Core: Forecast demand for [SKU group, locations group, time range]
- Choose your slice thickness in each of these 3 dimensions
- Thin Slice (local effects): Little data, highly uncertain forecasts
- Thick Slice (macro effects): Dependent on many external factors
- Hierarchical forecasting blends macro effects with local effects
- Good forecast captures structural patterns and regime shifts

Similar/Substitutable Products identified with *Embeddings*

- Demand Forecasting for similar/substitutable set of Products
- Can we automate identification of similarity/substitutability?
- Similarity by Visuals or by Descriptions or by Customer Interest
- Throw this heterogeneous data into deep neural network learning
- Deep inside the neural network, we find encodings of these products
- Similar/Substitutable products have similar encodings
- The technical term for these encodings is *Embeddings*
- Embeddings are low-dimensional numerical representations (*Vectors*)
- Capturing the most important features of products
- Captures various relationships between products, eg: complementarity
- Based on product images, descriptions, customer interest
- Embedding vectors have powerful algebraic/geometric properties
- Embeddings can be used as ML features for transfer learning

Book Embeddings flattened to 2-D Vectors



Transformers for learning the old and the new

- Forecasting needs to be "Predictive" as well as "Reactive"
- *Predictive:* Capture temporal patterns from long history
- *Reactive:* Pay attention to recent shifts and trends
- Transformer Networks eating the lunch of good old time-series stats
- The core methodology is a technique called *Self-Attention*
- It automatically identifies the relative importance of old versus new
- Transformers are accurate and fast (highly parallelizable)
- Have found great success in Vision and Text applications
- Now being ported to numerical time-series applications
- Embeddings blended with Transformers $\Rightarrow$ Heady Cocktail
- Not easy to pull off, requires considerable experimentation/tuning

Capturing Stock Price Cycles/Trends with Transformers

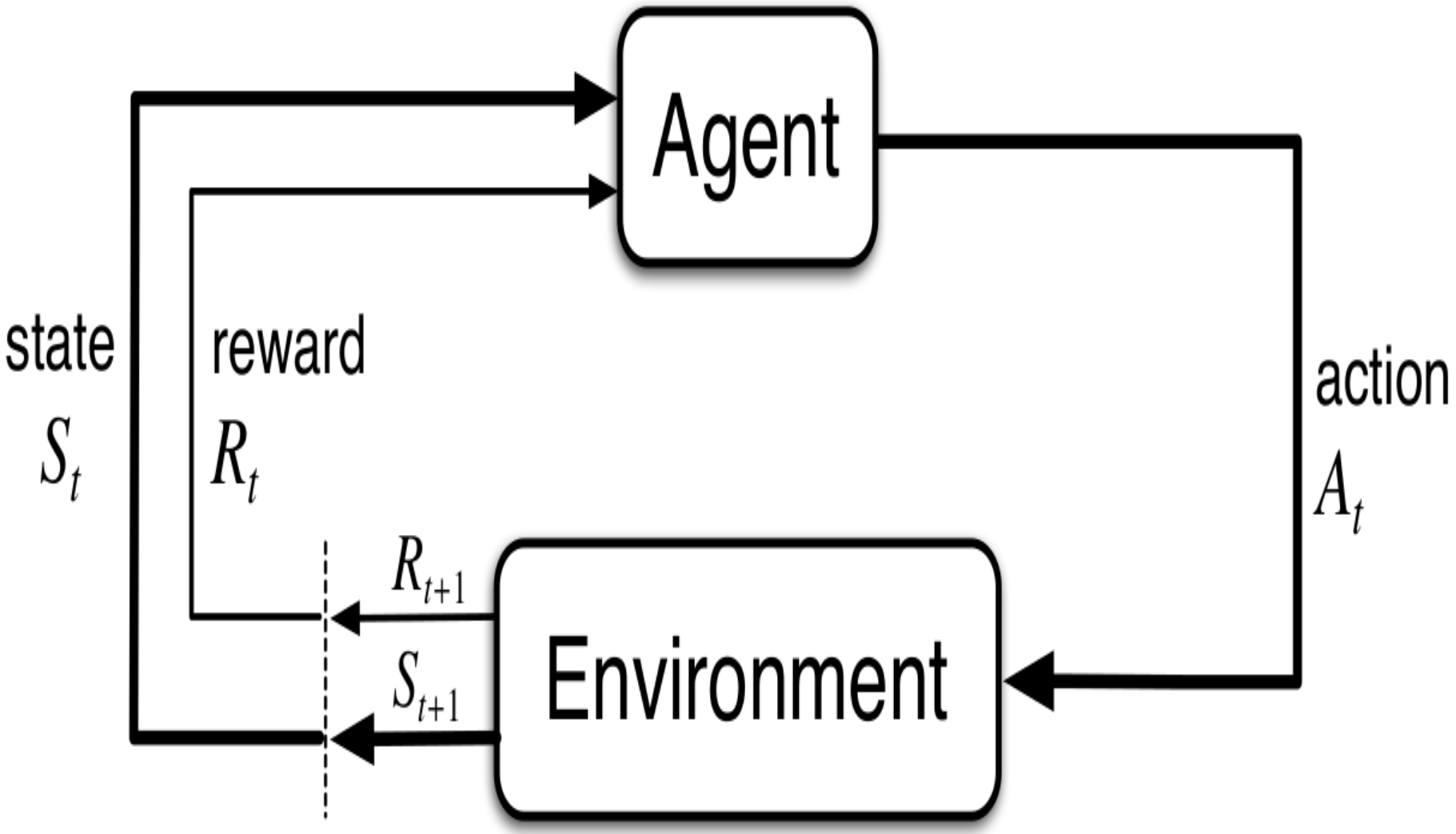
Inventory Replenishment

- Let's consider a simple example of a single SKU in a store
- The store experiences uncertain daily customer demand
- The store can order daily from a supplier carrying infinite inventory
- Cost associated with ordering, and order arrives in a few days
- Inventory in the store incurs a daily *Holding Cost* (per unit)
- Out-of-Stock incurs a *Stockout Cost* (per unit)
- There's a notion of *Ordering Cost* (Labor/Transport)
- When should one order? And in what quantity?
- Traditional OR literature focused on closed-form "solutions"
- Approx closed-form solutions are often used in real businesses
- In spite of the limitations of missing out on key real-world features

Welcome to the Real-World

- Real-world not so simple - frictions, constraints, uncertainties
- Supply can be constrained/uncertain
- Space and Throughput constraints
- Pricing/Marketing has a big influence on demand
- New products, substitutable products complicate matters
- In physical stores, there are *Presentation-Minimum* requirements
- Often, products are shipped in casepacks
- Irregular/Uncertain order frequency and lead times
- Supply-Chain network might be *Multi-Echelon*
- Inventory Count/Location is often uncertain
- Cost of Damage, Theft, Misplacement, Spoilage
- End-of-Season/Obsolescence/Spoilage requires Clearance/Salvage
- All of this calls for a formal Stochastic Control framework

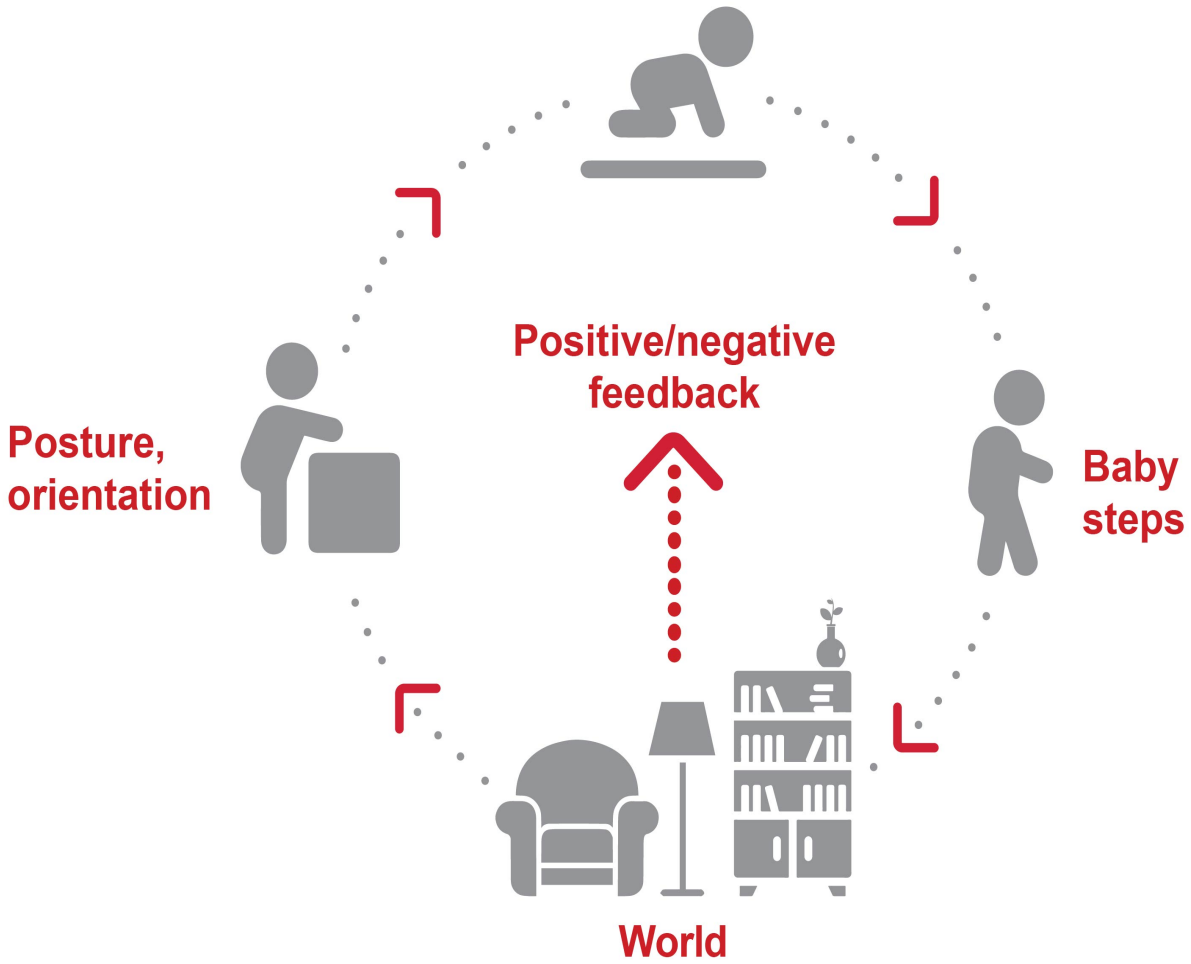
The Markov Decision Process Framework



Components of the MDP Framework

- The *Agent* and the *Environment* interact in a time-sequenced loop
- *Agent* responds to [*State*, *Reward*] by taking an *Action*
- *Environment* responds by producing next step's (random) *State*
- *Environment* also produces a (random) number we call *Reward*
- Goal of *Agent* is to maximize *Expected Sum* of all future *Rewards*
- By controlling the ($Policy : State \to Action$) function
- This is a dynamic (time-sequenced control) system under uncertainty
- MDP framework enables modeling real-world Replenishment problems

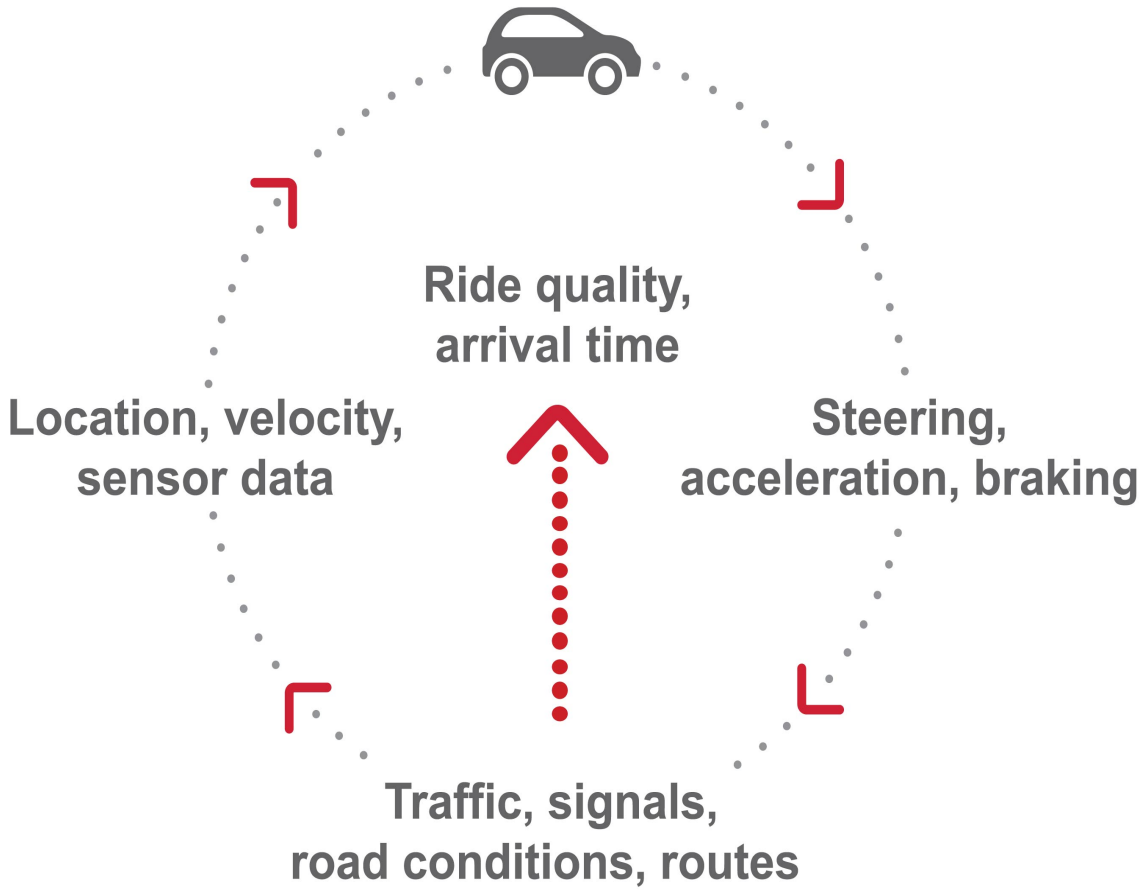
How a baby learns to walk



Many real-world problems fit this MDP framework

- Self-driving vehicle (speed/steering to optimize safety/time)
- Game of Chess (Boolean *Reward* at end of game)
- Inventory Replenishment to ensure high availability at low cost
- Make a humanoid robot walk/run on difficult terrains
- Manage an investment portfolio
- Control a power station
- Optimal decisions during a football game
- Strategy to win an election (high-complexity MDP)

Self-Driving Vehicle



Real-World Replenishment as a Markov Decision Process

- MDP *State* is current Inventory Level at the store
- *State* also includes current in-transit inventory
- *Action* is the multiple of casepack to order (or not order)
- *Reward* function involves all of the costs we went over
- State transitions governed by all of the uncertainties we went over
- Solve: Dynamic Programming or Reinforcement Learning
- Curse of Dimensionality and Curse of Modeling $\Rightarrow$ RL

How RL Works: Learning from Samples of Data

- RL incrementally learns from state/reward transitions data
- Typically served by a simulator acting as a *Simulated Environment*
- RL is a "trial-and-error" approach linking *Actions* to *Rewards*
- Try different actions & learn what works, what doesn't
- Deep Neural Networks are typically used for function approximation
- Big Picture: Sampling and Function Approximation come together
- RL algorithms are clever about balancing "explore" versus "exploit"
- Promise of modern A.I. is based on success of RL algorithms
- Potential for automated decision-making in many industries
- In 10-20 years: Bots that act or behave more optimal than humans
- RL already solves various low-complexity real-world problems
- RL has many applications in Suppy-Chain