AI Opportunities in Retail

Ashwin Rao

Stanford University

A bit about me

- Chief AI Officer at QXO (Distribution of Building Materials)
- Adjunct Professor, Applied Mathematics (ICME), Stanford University
- Past: VP of AI at Target Corporation
- Past: MD at Morgan Stanley, Trading Strategist at Goldman Sachs
- Leading Stanford's Mathematical & Computational Finance program
- Research & Teaching in: *RL and it's applications in Finance & Retail*
- Book: Foundations of RL with Applications in Finance
- Teaching faculty in HAI for retail companies
- Today I will talk about AI opportunities in Retail

Overview of AI Opportunities

- Deep Learning-based Demand Forecasting
- Operations: Inventory, Network, S&OP, Transportation, Logistics
- Commercial: Assortment, Presentations, Pricing
- e-commerce: AI Sales Agent, Personalization
- Employee Productivity: AI Agents serving as co-pilots/assistants

Demand Forecasting Requirements

- Forecasting at various granularities of ¡SKU, Location, Time¿
- Combine “predictive” versus “reactive” capabilities
- Understand SKU substitutability and complementarity
- Incorporating “tribal” knowledge into forecasting

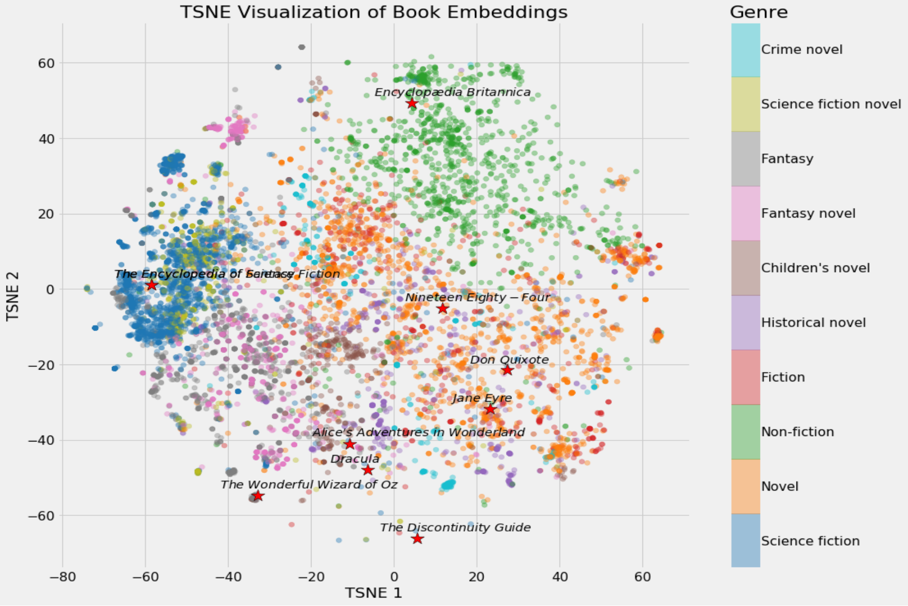
Deep Learning can deliver on these requirements

- Investment in Data Engineering
- Internal+External Data, Macro+Local Data, Real-time streaming
- Granular Forecasts targeted towards Operational Control
- Granular: Little data, highly uncertain forecasts, local factors
- Coarse Forecasts targeted towards Planning and Design
- Coarse: Depends on many external factors/macro factors
- Hierarchical forecasting blends macro effects with local effects
- "predictive" is core, "reactive" is an update upon event reception
- *Transformers* blend structural patterns with regime shifts
- *Embeddings* capture similarities and complementarities
- System for capturing "tribal" knowledge in data systems

Similar/Substitutable Products identified with *Embeddings*

- 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
- Embeddings can be used as ML features for transfer learning
- Embedding vectors have powerful algebraic/geometric properties

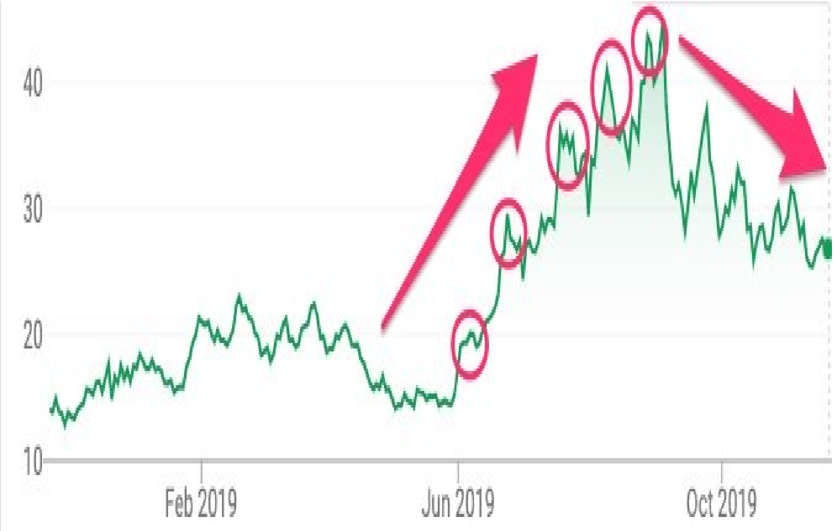
Book Embeddings flattened to 2-D Vectors



Transformers for learning the old and the new

- *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)
- Core Tech in LLMs
- Now being ported to numerical time-series applications
- Embeddings blended with Transformers ⇒ Practically Potent Combo
- Not easy to pull off, requires considerable experimentation/tuning

Capturing Stock Price Cycles/Trends with Transformers



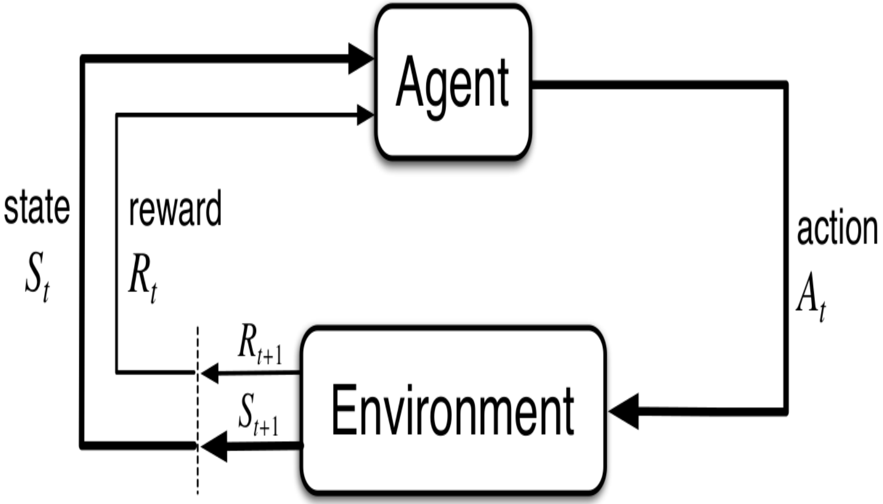
Inventory Replenishment

- 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 Decision Control framework

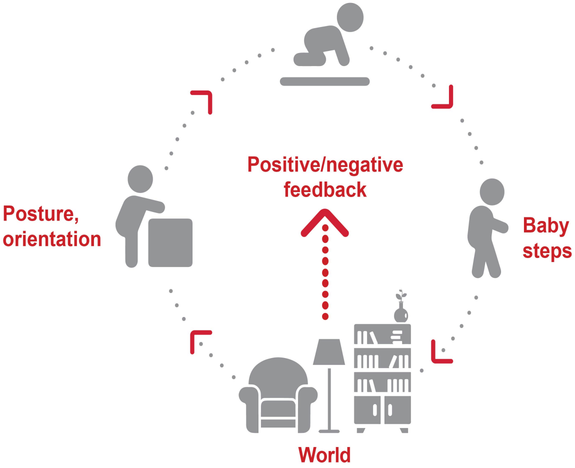
The Markov Decision Process Framework



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 ($Policy : State \rightarrow Action$) function
- This is a dynamic (time-sequenced control) system under uncertainty
- MDP framework enables modeling real-world Replenishment problems

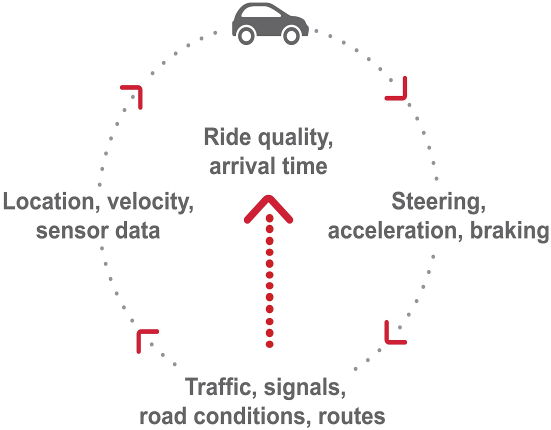
How a baby learns to walk



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 (covered in depth in my book)
- 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/warehouse
- *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 ⇒ 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
- RL has many applications in Suppy-Chain, more broadly in Operations
- RL covered in great detail in my book (theory, code and applications)