Reinforcemnet Learning Project Report

Q-Grocery Shopping Solution

Gaurav Kukreja

Table of Contents

Abstract

Q-Grocery Shopping Solution is a helper for buying groceries. It helps people spend less money, travel less, and avoid empty shelves. The system learns which shops have the best prices and items. It is useful for busy workers, families, old people, and delivery workers. This report shows how the system works, the tools used, how it was made, and the results.

Introduction

Background

Grocery shopping involves constant trade-offs between cost, convenience, and availability. Traditional ways require checking many stores by hand. often leading to inefficiency, wasted time, and increased expenses.

Objectives

- Automate grocery shopping decisions.
- Minimize combined cost of items and travel.
- Use AI to handle changing prices and stock.
- Provide actionable recommendations via a user-friendly interface.

System Overview

Q-Grocery Shopping Solution is a reinforcement learning-based agent that autonomously selects shops to purchase items in an optimal order.

Target Users

- Busy Professionals & Parents
- Budget Conscious Families
- Elderly or Mobility Limited Individuals
- Delivery App Drivers (minimizes dead miles and failed pickups)

Unique Features

- Autonomous Q-Learning Brain: Learns the optimal strategy from experience.
- Triple Optimization: Minimizes item cost, travel distance, and stock risk.
- Memory of Shop Behavior: Remembers availability of items over time.
- Unified Interface: No app switching; Devs provides single step instructions.
- Future-Proof: Scalable to multiple shops, items, a DBMS and real-time APIs.

Methodology

System Workflow

1. User opens the web app.
2. The app shows shop prices, stock, and distances.
3. User sets training parameters (episodes, α, γ, ε) and trains the Q-Learning agent.
4. The agent iteratively explores possible actions, updates Q-table using rewards, and learns optimal strategies.
5. Post training, the system displays the learned policy, Q-table, and reward curves.
6. In simulation, user can follow the AI or choose shops manually.
7. The system updates cumulative rewards and provides real time feedback.

Q Learning Algorithm

Input Parameters:

- **episodes**: number of times the agent will run the full training (default 500)
- **steps**: maximum steps per episode (default 100)
- **alpha**: learning rate — how fast Q-values update (default 0.1)
- **gamma**: discount factor — importance of future rewards (default 0.9)
- **epsilon**: exploration rate — how often to take random actions (default 0.2)

Q-Learning Algorithm

```
1. Initialize Q-table
   - Make all possible states: (current_shop, (item1_bought,
item2_bought))
   - Set all Q-values to 0

2. For each episode:
   - Pick a random starting state
   - Set total_reward = 0

3. For each step:
   - If both items are bought, end the episode
   - Choose an action (next shop):
     - With chance epsilon → pick a random shop
     - Otherwise → pick the shop with the best Q-value
   - Use take_action(state, action) to get:
     - next_state
     - reward
   - Update the Q-value slightly based on the reward and next state
   - Add reward to total_reward
   - Set state = next_state

4. After episode ends:
   - Save total_reward

5. After all episodes:
   - Return Q-table and list of total rewards
```

Technology Stack

Category	**Tools / Libraries**
Programming Language	Python
Frameworks / Libraries	NumPy, Pandas, Matplotlib, Streamlit
Database	Not connected (currently simulation-only)
Development Tools	VS Code, Git/GitHub, Jupyter Notebook
AI / Algorithms	Q-Learning, Epsilon-Greedy Exploration

Module Description

Module 1 – rl_project.py

- Purpose: Core Q-Learning engine and environment simulation.
- Inputs: Episodes, α, γ, ε, shop prices, availability, distances.
- Outputs: Trained Q-table, reward history.
- Key Functions: get_reward(), take_action(), epsilon_greedy(), q_learning().

Module 2 – streamlit_app.py

- Purpose: Interactive UI for environment visualization, training, and simulation.
- Features: Shop tables, network graph, Q-table display, learned policy, reward curves, AI recommendations.
- Libraries: Streamlit, Pandas, Matplotlib, NetworkX.

Implementation Details

- The system simulates 4 shops and 2 items with randomly generated prices, distances, and stock availability.
- During training, the Q-Learning agent explores possible actions, collects rewards (+50 per item bought minus cost and travel), and updates the Q-table.
- Streamlit provides a user interface to explore the environment, train the agent, and run simulations either manually or using Q learning.
- The agent gradually learns an optimal shopping strategy, balancing cost, distance, and stock risk.

Conclusion & Future Scope

Q-Grocery Shopping Solution shows a successful use of Q learning for real grocery shopping. The model learns the best strategies by itself, cuts shopping time and cost, and adapts to changing stores and stock.

Future Enhancements:

- Integrate real-time price and stock APIs.
- Support for multiple items and large shop networks.
- Integrate a database (DBMS) to store shop data, user history, and agent learning progress for better scalability and persistence.
- Advanced RL algorithms for faster convergence and scalability.

References

1. Sutton, R.S., & Barto, A.G. (2018). Reinforcement Learning: An Introduction. MIT Press.

2. Streamlit Documentation – Interactive Python Web Apps.

3. NumPy, Pandas, Matplotlib official documentation.