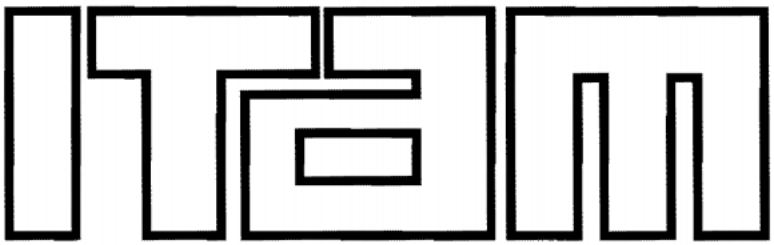
INSTITUTO TECNOLÓGICO AUTÓNOMO DE MÉXICO



Aprendizaje Reforzado para el Juego de la Distribución de Cerveza

T E S I S

QUE PARA OBTENER EL TÍTULO DE

MAESTRA EN CIENCIA DE DATOS

P R E S E N T A

MARÍA FERNANDA ALCALÁ DURAND

ASESOR: DR. ADOLFO JAVIER DE UNÁNUE TISCAREÑO

MÉXICO, D.F 2018

Con fundamento en los artículos 21 y 27 de la Ley Federal del Derecho de Autor y como titular de los derechos moral y patrimonial de la obra titulada "**Aprendizaje Reforzado para el Juego de Distribución de Cerveza**", otorgo de manera gratuita y permanente al Instituto Tecnológico Autónomo de México y a la Biblioteca Raúl Baillères Jr., autorización para que fijen la obra en cualquier medio, incluido el electrónico, y la divulguen entre sus usuarios, profesores, estudiantes o terceras personas, sin que pueda percibir por tal divulgación una contraprestación.

María Fernanda Alcalá Durand

_____

FECHA

_____

FIRMA

Agradecimientos

En este momento, le agradezco a Drake por hacer música tan espantosamente repetitiva: mi cerebro lo toma como ruido blanco y puedo concentrarme muy bien.

Tengo que pensar en un agradecimiento bonito para mi mamá, familia y asesor.

Abstract

Este trabajo de Tesis propone representar el conocido *Problema de la Distribución de la Cerveza* como un modelo multiagente de cadena de suministro, añadiendo una restricción tal que se aproxime más a un problema de la vida real: la producción de materia prima es finita y solamente ocurre en un periodo específico. El acercamiento elegido fue la aplicación de dos métodos de aprendizaje reforzado: *Policy Iteration* y *Q-learning*; se encontró que ambas técnicas convergen rápidamente a un óptimo. Este trabajo tiene dos implicaciones principales: la primera es que el aprendizaje reforzado no solamente es eficiente, sino suficientemente flexible como para identificar cambios permanentes en la demanda y adaptar las consecuentes políticas de compra; y la segunda es que, en subsecuentes trabajos, es necesario añadir restricciones adicionales al *Problema de la Distribución de la Cerveza* para obtener modelos más fieles a las condiciones del mundo externo y, por lo tanto, utilizables en un contexto real.

Índice general

Capítulo 1

Introducción

Necesito una cita cool para empezar mi tesis.

Fleo

Una cadena de suministro es el proceso en el cual una materia prima es transformada en un producto y este es vendido a un consumidor final [missing reference]. El comportamiento de una cadena puede representarse como un modelo en el cual existen flujos, ciclos y agentes. Por ejemplo, en una simplificaciøñ de producción automotriz, se representa a cada eslabón como un agente (la fábrica de partes, la ensambladora, el taller de pintura y la agencia de venta), los flujos como las entradas y salidas de materia en cada eslabón (por ejemplo, la fábrica de partes tiene como insumos metal y plástico, mientras que el taller de pintura tiene como insumo el auto ensamblado y litros de pintura) y los ciclos pueden depender de estacionalidad, disponibilidad o simple predisposición.

El estudio de las cadenas de suministro cubre un campo vasto, dado que existen una infinidad de problemas relacionados a ellas: desde transporte, logística y manejo de inventario hasta optimización de la localización geográfica para cada uno de los eslabones[missing reference]. Sin embargo, una vez que la cadena está en funcionamiento, uno de las principales dificultades es que los agentes encargados de optimizar las estrategias de demanda y producción de cada eslabón solamente pueden tomar decisiones "dentro" de aquel en el que se encuentran, y no tienen información más allá de los

eslabones inmediatamente conectados. Así, la información acerca de la demanda del consumidor se va diluyendo en cada nivel, además de que las decisiones tomadas tienen repercusiones más allá del futuro inmediato.

Dado que el objetivo principal de cada agente es minimizar los costos al tiempo de maximizar los ingresos, cada uno de ellos debe tratar de inferir el patrón global por medio de información local bastante restringida. Volviendo al ejemplo sencillo de la producción automotriz, la fábrica de partes debe ordenar metal y plástico suficiente para producir y cubrir la demanda de la planta ensambladora, pero ambos eslabones producen para el objetivo final: el consumidor. Sin embargo, la planta no tiene ningún incentivo real para compartir con la fábrica la cantidad exacta de autos ensamblados que produce o que vende cada periodo al taller de pintura. Esto obliga a cada eslabón a contar solamente con datos limitados, además de que los datos de demanda que reciben obedecen al tiempo real y no tienen la oportunidad de repetir experimentos.

Un modelo computacional que se comporte suficientemente parecido al mundo real, en el que todos los demás eslabones tomen estrategias que también maximizarían sus beneficios podría dar una opción: el experimento es replicable tantas veces como sea necesario y cada eslabón puede conocer una estrategia óptima para una gran cantidad de demandas de consumidor posibles.

1.1. Acercamiento en el Presente Trabajo

En este trabajo se modelará el Problema de Distribución de Cerveza, *The Beer Distribution Game*, ampliamente utilizado y explicado por el Profesor Sterman [8] en la Escuela de Administración y Dirección de Empresas Sloan del MIT, para ilustrar el concepto de *efecto látigo*. Este efecto recibió tal nombre debido a que la varianza en la información acerca de la demanda real tiene el mismo comportamiento que un látigo: mientras más lejano se encuentra del origen (consumidor), más amplia es la onda (varianza).

Existen acercamientos anteriores a este problema, en específico Chaharsooghi et al. [3] propone

ya un acercamiento con *Q-learning* [1] pero sin restricción de estacionalidad en la materia prima, Strozzi et al. [9], por medio de Algoritmos Genéticos,] y Zarandi et al. [11] proponen híbridos de un algoritmo genético y aprendizaje reforzado.

Por otro lado, algunas líneas de investigación como Busoniu et al. [2] se han concentrado en el aspecto de teoría de juegos que compete a este problema: si en un vendedor no tiene existencias, los consumidores podrán elegir otro. En este caso, los agentes son construidos como adversarios. Dado que en el modelo que se resuelve en el presente trabajo los agentes no son ni adversarios ni cooperativos, sino maximizan su propia utilidad, no se implementará como este tipo de modelo.

Sin embargo, todos los acercamientos anteriores suponen que los campos pueden adaptarse de inmediato a la demanda de la fábrica, de cierta manera tienen "producción infinita". El aporte de este trabajo es la imposición de restricciones basadas en tendencias catalogadas por el departamento de agricultura de los EUA en la producción de cebada, lo cual debería alterar el comportamiento de los agentes tal que mantengan una existencia positiva en sus almacenes para poder enfrentar la demanda en periodos de baja producción.

[1] Los nombres de los algoritmos serán representados con itálicas en este trabajo.

Capítulo 2

El Problema: Juego de Distribución de la Cerveza

The Beer Distribution Game [7], planteado por primera vez en la Escuela de Administración y Dirección de Empresas Sloan del MIT en los años 60 para ejemplificar el *efecto látigo*, llamado así por la similaridad que tiene el comportamiento la información en cada nivel de la cadena con el patrón ondulado que toma un látigo, en el cual existe una mayor amplitud de onda (comparable con el ruido o la varianza en la información) al alejarse del punto de origen (comparable al consumidor).

En su forma de juego de mesa, se presenta comúnmente a alumnos recién ingresados a distintas escuelas de negocios. El MIT publicó un artículo noticiero corto de Dizikes [4] en el cual se nota que, independientemente del rol que jueguen y de cuánta experiencia y preparación en negocios tengan, los humanos consistentemente fallan en encontrar la estrategia para maximizar la utilidad. Asimismo, nota que es inevitable la frustración, y que incluso los equipos que obtienen los mejores resultados del juego terminan lejos del óptimo.

El *Beer Distribution Game* ejemplifica la relación causal entre la toma de decisiones de cada agente con el comportamiento de todo el sistema, en este caso una cadena de suministro. Asimismo, presenta el efecto látigo claramente: el retraso en la información causa que, a través del tiempo,

el comportamiento de cada agente se vuelva menos constante cuanto más lejos se encuentre del consumidor final. Por último, evidencia las ineficiencias inherentes a tratar de resolver el problema enfocándose en los agentes, ignorando que es un sistema completo.

2.1. Principales características

El juego consiste, a grandes rasgos, en asignar a cada agente un rol en una cadena de suministro de cerveza, buscando maximizar las ganancias individuales al final del juego.

Existen cuatro jugadores: minorista (*retail*), mayorista (*wholesale*), distribuidor (*regional warehouse*) y fábrica (*factory*). Dentro del juego transcurren 52 semanas (un año), durante las cuales existe una demanda del consumidor, que se revela al inicio de la semana. De esta manera, el minorista debe cubrir (restringido a su inventario) la demanda del consumidor, y decidir la orden que pedirá al mayorista para recibir la siguiente semana. Cada jugador sigue instrucciones similares, con el objetivo de maximizar sus ganancias al final del juego.

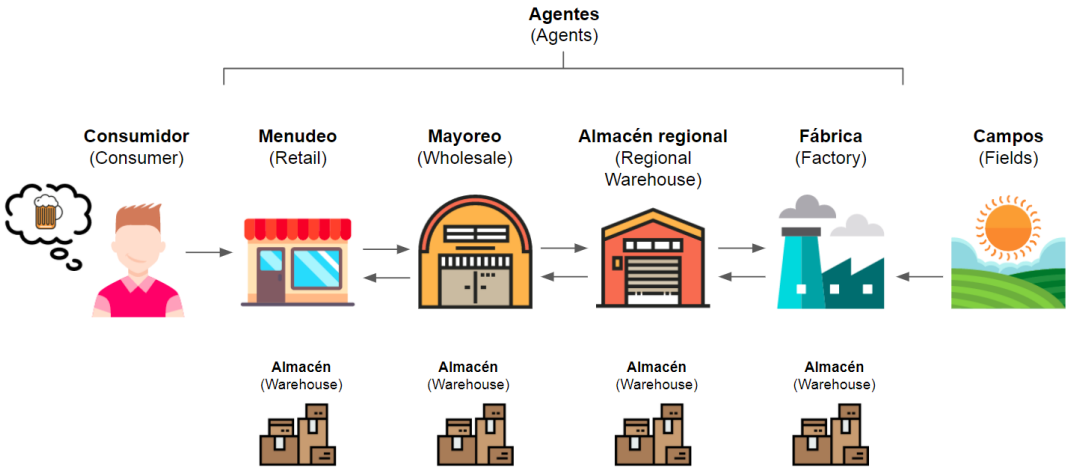
Todos los agentes cuentan con un inventario inicial de cajas de cerveza, y deben manejar correctamente su inventario para poder cumplir con la demanda del agente previo, al tiempo de minimizar los costos de almacenamiento por cada caja. Todos reciben ingresos por vender cajas de cerveza, e incurren en costos por comprar inventario, almacenar inventario, y por último, una penalización por no cubrir las órdenes. La estructura del juego se puede observar en la figura 2.1.[1]

De esta manera, las variables que tienen efecto en este problema son:

- Demanda del consumidor
- Producción (oferta) de los campos
- Precio de la cerveza
- Costo de almacenaje

[1] Iconos creados por Freepik, Iconnice, Roundicons en *www.flaticon.com*

Figura 2.1: Distribución de Cerveza



- Costo de oportunidad por órdenes no cumplidas
- Inventario inicial

Todas las variables, menos la demanda del consumidor y la producción en los campos, pueden ser declaradas para el sistema (igual para todos los agentes) o individualmente (diferente para cada agente). Como se verá más adelante, el sistema es sumamente sensible a pequeños cambios en cualquiera de las variables; por ejemplo, si un agente comienza el juego con suficiente inventario para todo el año, el agente superior nunca podrá venderle cerveza e incurrirá solamente en costos por almacenaje.

2.1.1. El Efecto Látigo

El *Efecto Látigo* es un fenómeno que se produce en cadenas de suministro. Se llama de esa manera porque, mientras más "arriba" en la cadena de suministro se encuentre un agente (es decir, más lejos del contacto directo con el comprador), más distorsionada es la información que tiene acerca de la verdadera demanda del comprador; tal varianza se puede visualizar como una curva que asemeja un látigo. Chaharsooghi et al. [3] nota que la distorsión se debe mayormente a tres factores: prejuicios en la información de la demanda por parte de los miembros cercanos al consumidor, retraso en el intercambio de información entre los miembros de la cadena y soporte logístico

inapropiado a través de la cadena.

Se puede ejemplificar con el siguiente escenario:

1. El consumidor, que generalmente compra 6 cervezas, ahora quiere 10, pero la tienda minorista solamente cuenta con 7. El minorista le venderá todo su inventario, pues es la acción que maximiza su ganancia. Después, debe decidir si volverá a tener un inventario de 6 o si debe pedir un número mayor de cervezas, atendiendo la posible creciente demanda. Decide pedir 9 cervezas al siguiente nivel, la tienda de mayoreo.

2. El mayorista cuenta con 17 cervezas. Llena el pedido del minorista, pero decide que tenía guardado demasiado inventario, así que se queda con 8 cervezas en su almacén, sin hacer una orden al siguiente nivel, la tienda de distribución.

3. La tienda de distribución decide no comprar unidades a la fábrica, dado que no disminuyó su inventario.

4. La fábrica conoce la restricción de estacionalidad de la cebada, así que compra una mínima cantidad a los campos.

En este escenario, el mayorista obtuvo información distorsionada acerca del repentino crecimiento en la demanda del comprador, mientras que la tienda de distribución podría incluso interpretar que el comprador disminuyó su consumo. Si este comportamiento se mantiene durante algunos periodos más, recibiría la noticia (por medio de un incremento en las órdenes regulares) con un retraso considerable.

En el caso de que la demanda sea determinista (con un castigo por órdenes no cumplidas), la orden óptima en cada tiempo para cada miembro de la cadena es “una por una”, es decir, demandar al agente superior exactamente lo mismo que fue demandado por el agente inferior.

2.1.2. La dinámica de las existencias y flujos

Las existencias y flujos (*stocks and flows*) son componentes básicos en cualquier sistema dinámico [2]. Las existencias son acumulaciones basadas en la diferencia entre los flujos entrantes y los salientes. El flujo neto a una existencia en un tiempo determinado t es la tasa de cambio de la existencia en el tiempo t:

$$Existencia(t) = \int_{t_0}^{t} [Entrada(s) - Salida(s)]ds + Existencia(t_0)$$

Es común que en un sistema con varias existencias con sus respectivos flujos ocurran retrasos. Esto sucede porque, en general, los flujos de entrada y salida son gobernados por decisiones diferentes. En el Juego de Distribución de Cerveza, para cada agente, su flujo de salida depende de la demanda del agente inmediatamente inferior limitada por su propia existencia, y su flujo de entrada depende de su propia demanda limitada por la existencia del agente inmediatamente superior. Esto, aunado a un comportamiento ligeramente aleatorio de la demanda del consumidor, puede desembocar en insuficiencia de existencias para cualquier agente en cualquier tiempo t.

Un capital está en equilibrio cuando no cambia (un sistema está en equilibrio cuando ninguna de sus existencias cambia). Esta condición se obtiene cuando el cambio neto de sus flujos es cero, es decir, cuando el flujo entrante es igual al flujo saliente.

La solución trivial para este equilibrio es que todos los flujos sean nulos (*equilibrio estático*, sin embargo, en el caso de la cadena de suministro, este escenario llevaría a que ningún agente tuviera ventas, y por consiguiente, tampoco tuviera ganancias. Una situación más adecuada sería un *equilibrio dinámico*, en el cual las existencias totales se mantienen constantes, pero los flujos son positivos. Para la cerveza, esto significaría que la cantidad de cerveza en los almacenes se mantiene constante, pero cada caja de cerveza se mueve de un agente a otro en algún punto en el tiempo. Sin restricciones de oferta, la situación ideal tendería a no guardar cerveza en los almacenes. Sin embargo, el efecto látigo complica la estimación de los flujos entrantes y salientes en cada tiempo,

[2] Para mayor profundidad acerca de sistemas dinámicos y el comportamiento de las existencias y flujos, Sterman [8] profundiza y provee de las bases adecuadas para sistemas más complejos.

pues para cada agente son desconocidas tanto la demanda del agente inferior como la capacidad del agente superior de cubrir la demanda propia. Existe una complicación extra al añadir estacionalidad en la producción de los campos de cebada.

Capítulo 3

Aprendizaje Reforzado

3.1. Orígenes y Descripción

Su principio se basa en la psicología conductista: un *agente* busca ser recompensado por un premio, el cual obtiene cuando realiza una secuencia de *acciones* que lo llevan a concluir una tarea exitosamente. Además, para maximizar la cantidad de *recompensa* que recibe - o, alternativamente, minimizar el tiempo que espera entre un premio y el siguiente - comienza a optimizar su política (π) para llegar a la meta satisfactoriamente. Una explicación mucho más profunda de las bases e historia del aprendizaje reforzado, así como la mayor parte de las definiciones mencionadas en este capítulo, pueden ser encontradas en Sutton y Barto [10].

3.1.1. Terminología común

Existen conceptos utilizados en prácticamente la totalidad de los algoritmos de aprendizaje reforzado, independientemente de las particularidades que presenten. A continuación se presenta una lista, basada en la recopilación de [5] no exhaustiva de ellos, pero que cubre los términos utilizados en este trabajo.

1. Agente: es el concepto más abstracto, pues es la entidad que aprenderá durante el proceso. Por ejemplo, puede ser un ratón robótico aprendiendo a llegar al queso en el laberinto; o un

jugador virtual aprendiendo a superar todos los niveles de un videojuego.

2. Estado: describe la situación en la cual se encuentra el agente. Por ejemplo, para el ratón robótico, el estado es la casilla en la que se encuentra. El estado puede tener tantas dimensiones como sea necesario: por ejemplo, el estado de un coche autodirigido sería descrito por su posición, velocidad, objetos en su radar e incluso cantidad de gasolina en el tanque.

3. Mundo: se refiere la totalidad de los posibles estados.

4. Acción: es lo que un agente puede hacer en cada estado. Generalmente, el conjunto de acciones que puede tomar un agente es finito, aunque el conjunto de acciones que tome durante todo el tiempo pueda tener combinaciones que se aproximen a infinito. Por ejemplo, para el ratón robótico, sus acciones son realizar un movimiento hacia adelante, atrás, derecha o izquierda.

5. Recompensa: cuando un agente toma una acción en un estado, recibe una recompensa. Es muy importante notar que la recompensa puede tomar muchas formas: es posible que sea solamente la meta última en vez de existir después de cada acción (como el queso para el ratón). La recompensa también puede ser negativa, equivalente a un castigo, lo cual causa que el agente aprenda a evitar las acciones que lo llevan a ella.

6. *policy* es una función que toma como argumento un estado s y devuelve una acción a; equivalente a la estrategia del agente. La meta del aprendizaje reforzado es encontrar la *policy* óptima. Generalmente se le representa como $\pi(s) -> a$.

Una vez que tenemos asignados estos cuatro elementos, resulta relativamente simple explicar el problema, pero no así la implementación. Por ejemplo, para el ratón solamente recibirá la recompensa después de un movimiento específico que lo lleve a la casilla en la que se encuentra el queso, todos los demás movimientos son solamente un medio para acercarse. El aprendizaje reforzado resuelve esta situación concentrándose en las recompensas a largo plazo en vez de las inmediatas. Si descontamos el valor del queso por cada casilla que le toma llegar a él, entonces buscará el camino más corto, optimizando su secuencia de acciones interactuando con el mundo que lo rodea para descubrir relaciones acción-estado.

La principal característica del agente es que tiene la capacidad de tomar decisiones sobre sus acciones, las cuales son su forma de interactuar con el *mundo*, llevándolo de un *estado* a otro. El agente no tiene acceso a todas las consecuencias de sus acciones; de hecho, ni siquiera conoce todo el mundo.

El agente toma una acción en el tiempo t, la cual depende del estado s_t del mundo. En $t+1$, el mundo reaccionó ya a la interacción del agente con él, así que el agente recibe una recompensa r_{t+1} y toma una nueva acción dependiendo del estado s_{t+1} del mundo. Sin embargo, no es óptimo seleccionar acciones solamente con base en la recompensa r_{t+1}, pues la naturaleza temporal del problema lo convierte en un problema a largo plazo, y el agente estaría considerando solamente consecuencias en el corto plazo.

Así, el agente debe aprender que existe un *retraso* entre cada acción que toma y el premio. Supongamos que, en una cuadrícula, el premio se encuentra en la casilla (x,y). El agente solamente puede llegar a esa casilla meta desde las adyacentes, pero si no se encuentra en una de estas, primero debe acercárseles. Así, cuando el agente comienza su exploración, irá aprendiendo que, lejos de que la recompensa sea inmediata, debe tomar una secuencia de acciones para llegar a ella.

Podemos entonces definir la *función de valor* asociada a la política como el valor esperado de la recompensa al tiempo t dado que el agente se encuentra en el estado s.

Uno de los conceptos más importantes en el aprendizaje reforzado es la *Ecuación de Bellman*. En esta sección se presenta el desarrollo de ésta.

La idea básica es que la recompensa en el tiempo t es:

$$R_t = r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \gamma^3 r_{t+4} + ...$$
$$R_t = r_{t+1} + \gamma \left(r_{t+2} + \gamma r_{t+3} + \gamma^2 r_{t+4} + ... \right)$$
$$R_t = r_{t+1} + \gamma R_{t+1}$$

El valor de un estado s se puede definir como la recompensa esperada dado que el agente empieza en ese estado, y depende de su *policy*:

$$V^{\pi}(s) = E_{\pi}\left\{R_t | s_t = s\right\} = E_{\pi}\left\{\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} | s_t = s\right\}$$

También es necesario que el agente ajuste su comportamiento mientras transcurre el tiempo: al principio debe explorar para conocer la mayor cantidad de consecuencias a sus acciones posibles, pero debe mantener el conicimiento de cuáles acciones le han reportado buenas acciones y tomar esas decisiones más seguido. A esta estrategia de exploración se le llama $\epsilon - greedy$.

Definamos $p_r t$ y $p_t t$ como las probabilidades al tiempo t de exploración y explotación, respectivamente. Entonces:

$$p_r t = 1 - \epsilon(t)$$
$$p_t t = 1 - p_r t \qquad \forall t$$

La función ϵ suele ser implementada como decreciente de forma lineal para el aprendizaje, de tal forma que mientras pasa el tiempo, el agente escoge las acciones conocidas que le reportan mayor utilidad más seguido; junto con un parámetro ρ aleatorio para asegurar que siempre existe una probabilidad positiva de explorar.

Generalmente se supone este tipo de problemas como Procesos de Decisión de Markov (MDP), cuya principal característica es que cumplen con la famosa propiedad de Markov: a grandes rasgos, el futuro solamente depende del presente, no del pasado.

Formalmente, un proceso de Markov es una tupla (X, U, f, ρ) en donde X es el conjunto finito de estados, U es el conjunto finito de aciones de un agente, $f : X * U * X \to [0, 1]$ es la función de probabilidad de transición de estado, y $\rho : X * U * X \to \mathbb{R}$ es la función de recompensa.

Cuando el conjunto de acciones (o la *policy*) a tomar es difícil de aprender porque no tenemos ejemplos, o el mundo / conjunto de acciones / conjunto de consecuencias es demasiado grande, es

apropiado utilizar Aprendizaje Reforzado en lugar de Aprendizaje de Máquina regular. Gracias a que el agente va descubriendo y acercándose a una *policy* óptima, no es necesario que explore todas las acciones posibles.

En este trabajo aplicaremos dos algoritmos sumamente conocidos y aplicados en un sinfín de ámbitos: *Policy Iteration* y *Q-learning*, y compararemos su rendimiento al aplicarlos al problema de la distribución de cerveza.

3.2. Policy Iteration

Se empieza con una *policy* aleatoria, se encuentra el valor de ella, y se realiza un paso que encuentra una *policy* nueva (mejor) basado en la anterior. De esta manera, *policy iteration* asegura convergencia a la *policy* óptima,

La ecuación que representa este algoritmo es:

$$\pi_0 \xrightarrow{E} V^{\pi_0} \xrightarrow{I} \pi_1 \xrightarrow{E} V^{\pi_1} \xrightarrow{I} \pi_2 \xrightarrow{E} ... \xrightarrow{I} \pi_* \xrightarrow{I} V^{\pi_*}$$

En donde $\xrightarrow{E}$ significa que una *policy* se *evalúa*, y $\xrightarrow{I}$ significa que se mejora (por *improvement*, en inglés).

Se suele preferir este algoritmo sobre *value iteration*, pues es común que se encuentre la *policy* óptima mucho antes de que la función de valor converja. Además, dado que se itera sobre la *policy* anterior y se busca una mejor, el algoritmo converge rápidamente y cada paso asegura una mejora.

3.3. Q-Learning

w Uno de los algoritmos más populares de aprendizaje es *Q-learning*. Obtiene este nombre por la *función Q*, el cual indica el valor descontado de la recompensa asociado a una acción en un estado determinado.

3.3.1. Conceptos

Recordemos la ecuación de valor de un estado que fue desarrollada en capítulos anteriores:

$$V^{\pi}(s) = E_{\pi}\left\{R_t | s_t = s\right\} = E_{\pi}\left\{\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} | s_t = s\right\}$$

De aquí sigue que el valor de tomar una acción específica a en el estado s usando la *policy* π es:

$$Q_{\pi}(s,a) = E_{\pi}\left\{R_t | s_t = s, a_t = a\right\} = E_{\pi}\left\{\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} | s_t = s, a_t = a\right\}$$

$$V(s) = \max_{a} Q(s,a)$$

$$Q(s,a) = R(s,a) + \gamma * \max_{a} Q(s^{'}, a^{*})$$

Donde s' es el siguiente estado, y a^* representa todas las acciones posibles. Al estimar la función Q para cada par de estado con acción, es posible encontrar la mejor acción para cada estado y, así, obtener una política óptima.

Al iniciar el periodo de aprendizaje del algoritmo la función Q se establece como $Q(s,a) = 0 \forall s, a$, y en cada paso (e iteración) se actualiza su valor.

3.3.2. Algoritmo

1. Asignar $Q(s,a) = 0$ para todos los estados y acciones.
2. Posicionarse en un estado s
3. Seleccionar acción a^* y ejecutar
4. Recibir recompensa r
5. Observar estado nuevo $s^{'}$
6. Actualizar $\hat{Q}(s,a) = r(s,a) + \lambda \max_{a'} \hat{Q}(s^{'}, a^{'})$
7. Asignar nuevo estado $s \leftarrow s^{'}$

8. Volver a 2 hasta convergencia

3.4. Aprendizaje Reforzado Multi-Agente (ARM)

Existen muchos problemas que inherentemente pueden ser resueltos, pero es necesario que más de un agente aprenda de manera simultánea, especialemnte si la recompensa al final es compartida entre todos ellos. Por ejemplo, imaginemos que tenemos dos robots, ladrillos y un cuadro dibujado en el piso. Cada uno de los robots está cerca de esquinas opuestas y tiene acceso a la mitad de los ladrillos. La recompensa final se les otorga cuando forman un cuadrado de ladrillos, y es mayor mientras menor sea el tiempo que les tome terminar. En este caso, la estrategia óptima es que cada uno de los robots coloque, en cada tiempo, el ladrillo que más cerca le quede. El probema se vuelve sumamente complejo, pues cambios en la estrategia de un agente puedem afectar la estrategia de otros agentes. Por ejemplo, si uno de los robots tiene un comportamiento errático porque recibe un castigo si le sobra energía al terminar la tarea, el otro deberá adaptarse a esta nueva forma de actuar.

Muchas de las definiciones específicas del aprendizaje reforzado multi-agente, así como de su intersección con teoría de juegos, pueden encontrarse en Bloembergen y Hennes [1].

Existen pros y contras relacionados al aprendizaje reforzado multiagente (ARM), como los referidos por Busoniu et al. [2]. Algunos beneficios son que es posible que algunos agentes asuman tareas ajenas si esto ayuda a la optimización del premio final, es inherentemente rubusto, y generalmente están diseñados de tal manera que añadir nuevos agentes es fácil. Por otro lado, algunos retos son la escalabilidad debido al crecimiento exponencial de la dimensión del espacio estado-acción, o a la complejidad que se genera al añadir cada nuevo agente.

En este trabajo, consideraremos a cada eslabón de la cadena de suministro como un agente. Cada agente solamente puede comunicarse con los niveles inmediatamente vecinos; es decir, las únicas interacciones que puede tener con el mundo son el número de órdenes que recibe del nivel inferior y el inventario que pide al nivel superior. De esta manera, nuestros agentes no son adversarios, cooperativos ni independientes. el objetivo de cada agente es maximizar sus recompensas

individuales.

Sin embargo, como hemos definido una penalización por mantener cerveza en el inventario (el costo del almacén), la decisión concerniente a la petición del nivel inferior queda determinada: venderá todo lo que pueda, pues cada venta le reporta una ganancia, y no llenar la orden completa cuando tiene suficiente inventario lo haría incurrir en un costo innecesario.

Esto quiere decir que, para cada agente, el conjunto de **acciones** que puede tomar es solamente el número de cervezas que pedirá al nivel inmediatamente superior en cada tiempo t.

Por lo tanto, lo que tendrá guardado en la bodega en el tiempo t estará constituido por el número de cervezas que tenía en el tiempo anterior $t-1$, menos el número de cervezas vendidas, más el número de cervezas que recibe del nivel inmediatamente superior por el pedido de reaprovisionamiento, restringido a que cada agente solamente cubrirá la orden del nivel inferior si tiene suficiente inventario para hacerlo.

El objetivo de cada agente es maximizar su recompensa. Sin embargo, este es un problema ligeramente diferente a los comunes de *Q-learning*, en los cuales el valor de la recompensa es conocido y, una vez encontrado, se buscan las acciones óptimas "de atrás hacia adelante" (como el ejemplo típico de una cuadrícula).

Su política está definida con base en la función Q, una vez que el proceso de aprendizaje fue finalizado, de esta manera, puede realizar una búsqueda sobre todas las posibles acciones en los estados y sencillamente escoger la mejor, lo cual converge a la política (cuasi)óptima.

Es importante destacar que este sistema toma solamente una de las ramas que existen en la industria de cualquier producto (existe más de un minorista, etc.), e incluso, toma solamente un producto. Aún así, es un sistema complejo bastante robusto y sensible a cambios pequeños.

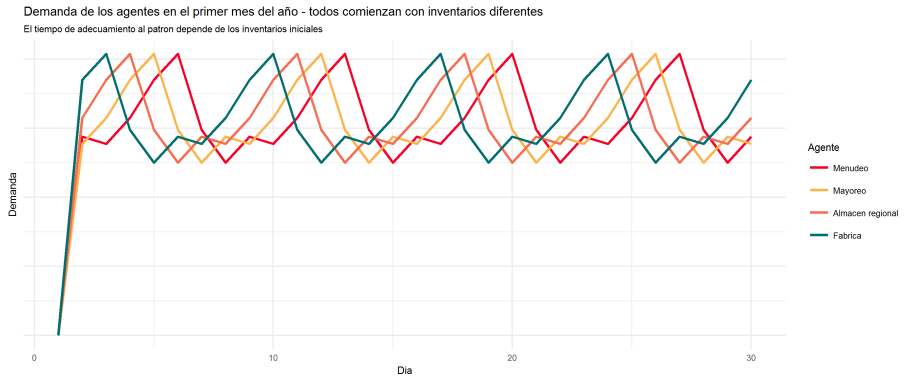
Capítulo 4

Acercamientos Previos a la Solución

4.1. Estática: máxima ganancia posible

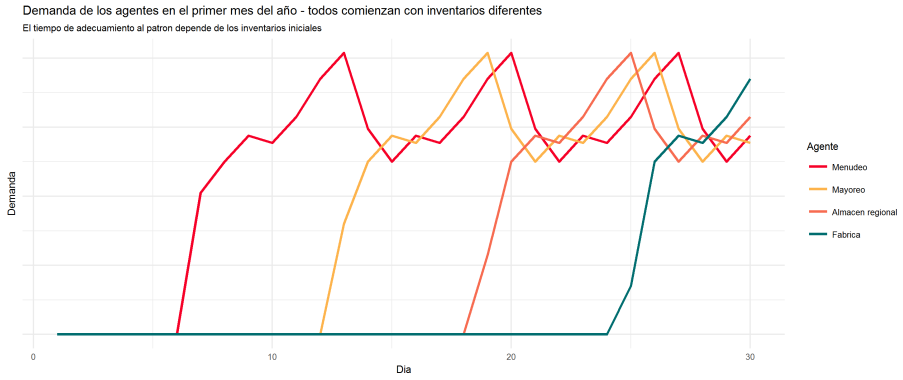
El sistema tiene una solución analítica si no existe una restricción en la oferta (es decir, el campo siempre puede cubrir la demanda de la fábrica) y cada agente tiene en el día n información acerca de la demanda del día $n+1$ de su agente inmediatamente inferior. En este caso, los agentes ni siquiera necesitan usar sus almacenes: sencillamente compran cada día lo necesario para el día siguiente. Podemos observar el comportamiento de esta solución en la figura 4.1.

Figura 4.1: Distribución de Cerveza



Sin embargo, incluso cuando todos los agentes tienen información perfecta, podemos observar una versión muy ligera del efecto látigo. Si el minorista tiene suficiente cerveza para cubrir algunos días de demanda del consumidor, entonces no tiene incentivo para comprarle al mayorista hasta que su inventario se agote. Esta situación puede replicarse en los dos niveles superiores, de tal manera que la fábrica no recibe nada de información acerca de la demanda del consumidor durante una cantidad considerable de tiempo. Este efecto se puede observar en la figura 4.2.

Figura 4.2: Distribución de Cerveza



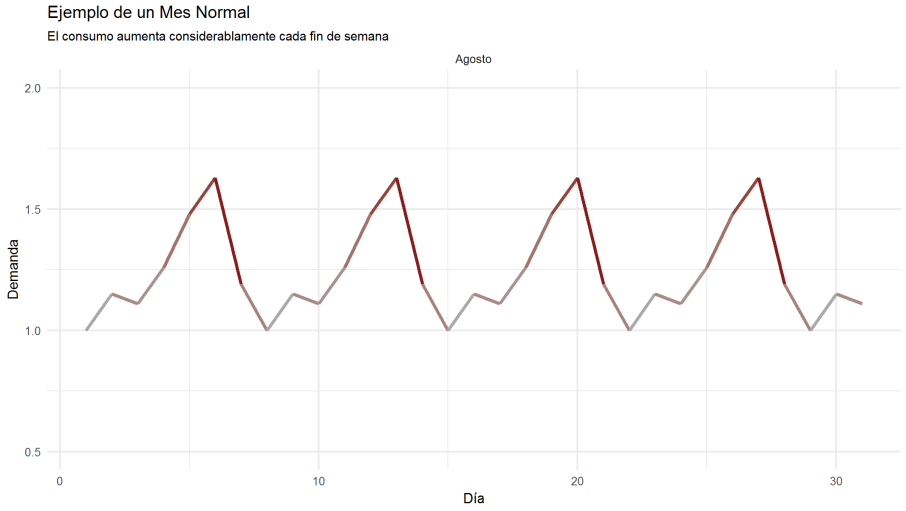
Si la fábrica quisiera estimar su demanda para el siguiente año solamente con esta información, tendría que crear un modelo (aunque fuera muy sencillo) para interpolar aquellos días en los cuales no tuvo información. En el caso optimista, usará el mismo patrón observado retroactivamente, y tendrá una aproximación relativamente buena. En el caso pesimista, supondrá que la demanda del consumidor durante esos días fue efectivamente cero, y sin duda causará una burbuja de falta de inventario que se propagará hacia abajo en la cadena de suministro. El efecto látigo habrá hecho de las suyas.

4.2. Estacional y aleatoria: más parecido al mundo real

En el mundo real, ningún producto tiene una demanda perfectamente constante. Más aún, existen una infinidad de productos cuya demanda varía con un patrón estacional: por ejemplo, las medicinas antigripales se venden más en invierno. La cerveza (y, en general, las bebidas alcohólicas) también siguen este tipo de patrones.

En primer lugar, existe un patrón semanal bastante esperado: según], se consume más cerveza los fines de semana (ver la figura 4.3). Además, existe un patrón relacionado a las festividades comunes. Según], el consumo de bebidas alcohólicas se duplica en las festividades navideñas.

Figura 4.3:



Para el presente trabajo, combinaremos los dos patrones descritos anteriormente, y añadiremos un pequeño componente aleatorio que cambiará la demanda cada año. El comportamiento "base" de la demanda de puede consultar en la figura 4.4. Un ejemplo del comportamiento perturbado por un poco de aleatoriedad se puede consultar en la figura 4.5. Sin embargo, cabe mencionar que este es un modelo simple, pues en algunas regiones, podría también existir un patrón relacionado al clima, o incluso un alza en la demanda siguiendo temporadas deportivas.

Como el comportamiento varía a lo largo del año y existe un costo por almacenar inventario, cada agente debe prepararse con adecuada anticipación para los picos de demanda. Además, el componente aleatorio resulta en la necesidad de conservar inventario de seguridad (en inglés, *safety stock*) para asegurar que la demanda será cubierta la mayor parte del tiempo.

Figura 4.4:

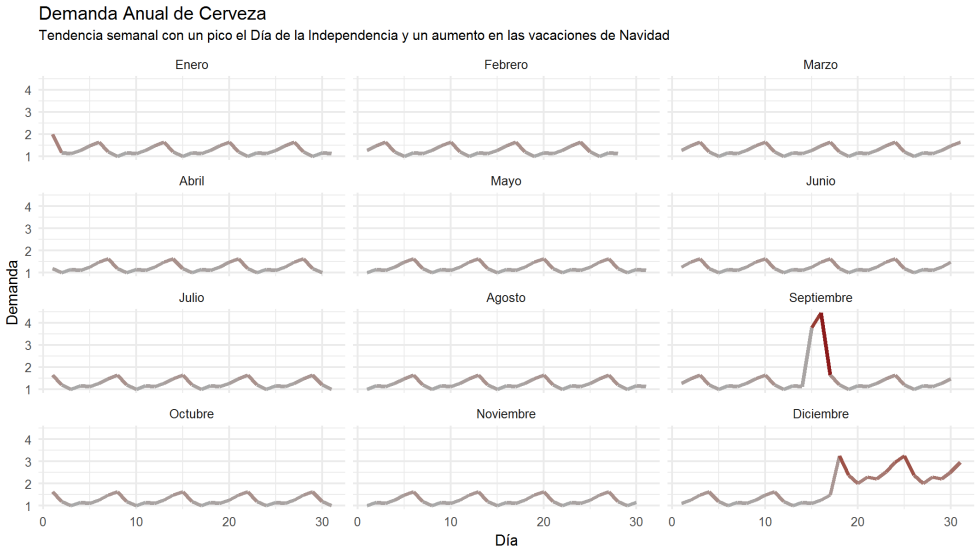
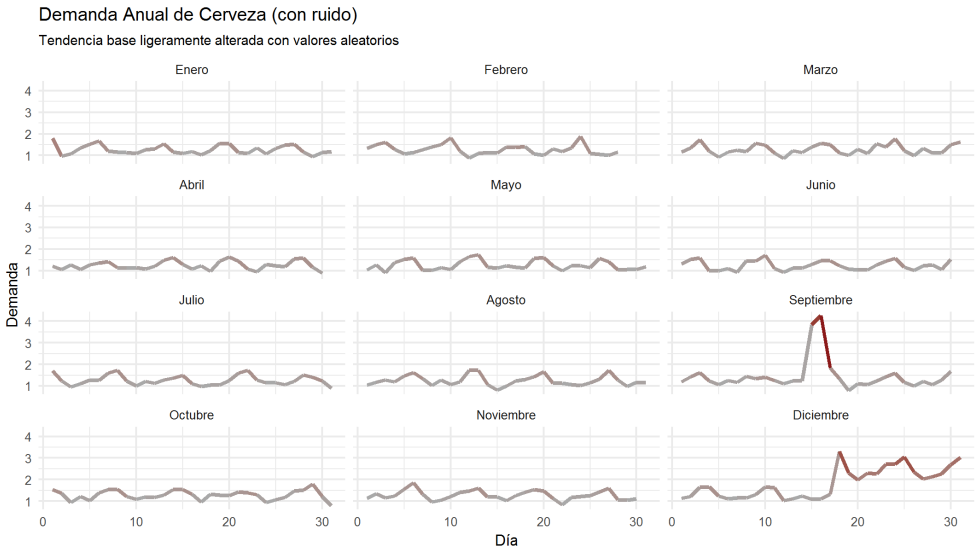


Figura 4.5:



4.3. Q-Learning

Chaharsooghi et al. [3] publicó ya un acercamiento a este problema usando *Q-learning*, sin embargo la solución que encuentra implica que los inventarios (al final del día, después de las transacciones del día) se aproximan a cero, consecuencia directa de que los agentes minimizan su costo

de inventario. Además, la demanda del consumidor obedece un patrón aleatorio.

Sin embargo, ninguna de estas soluciones considera dos factores importantes de la vida real: la temporalidad de la demanda (por ejemplo, para la cerveza la cantidad demandada es mayor durante el fin de semana) y la restricción de temporalidad de producción de materias primas. En el mundo agrícola, los campos solamente producen ciertas cosechas en ciertos periodos, y es necesario tener suficiente inventario para cubrir la demanda durante todo el año. En el siguiente capítulo, se presentará este escenario como un nivel extra de complejidad al modelo, limitando la oferta de cebada por parte de los campos.

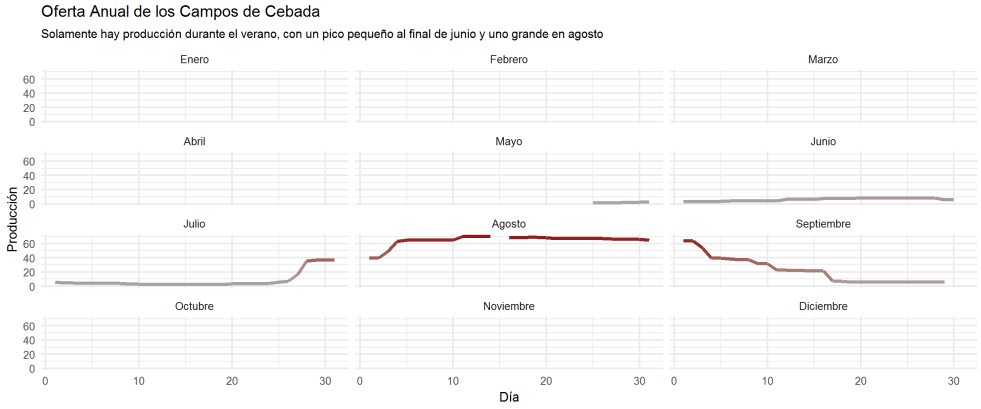
Capítulo 5

La Complicación: Restricción de Estacionalidad en los Campos

Todas las soluciones que se han propuesto anteriormente suponen que el agente al final de la cadena (el proveedor de materias primas) tiene inventario infinito. Sin embargo, en la vida real esta situación no se presenta: especialmente en materias primas que provienen de la siembra, existen ciclos de siembra y cosecha. Para construir nuestro modelo de la manera más aplicable a la realidad posible, podemos agregar la restricción correspondiente: el productor de materias primas solamente lo hace en ciertos momentos del año.

Utilizando los datos más recientes disponibles del departamento de agricultura de los Estados Unidos de América (ver of Agriculture [6]) hemos obtenido los ciclos naturales de uno de los principales componentes de la cerveza, la cebada, los cuales se pueden consultar en la figura 5.1. Podemos observar que la producción es nula entre octubre y mayo, con la mayor parte concentrada en agosto y comienzos de septiembre. Dependiendo de la magnitud de la demanda comparada con la oferta, esto podría significar que la cadena de suministro tiene que surtirse de cebada durante este periodo para poder cubrir la demanda del resto del año, haciendo uso de la capacidad de almacenamiento en sus respectivos almacenes.

Figura 5.1:

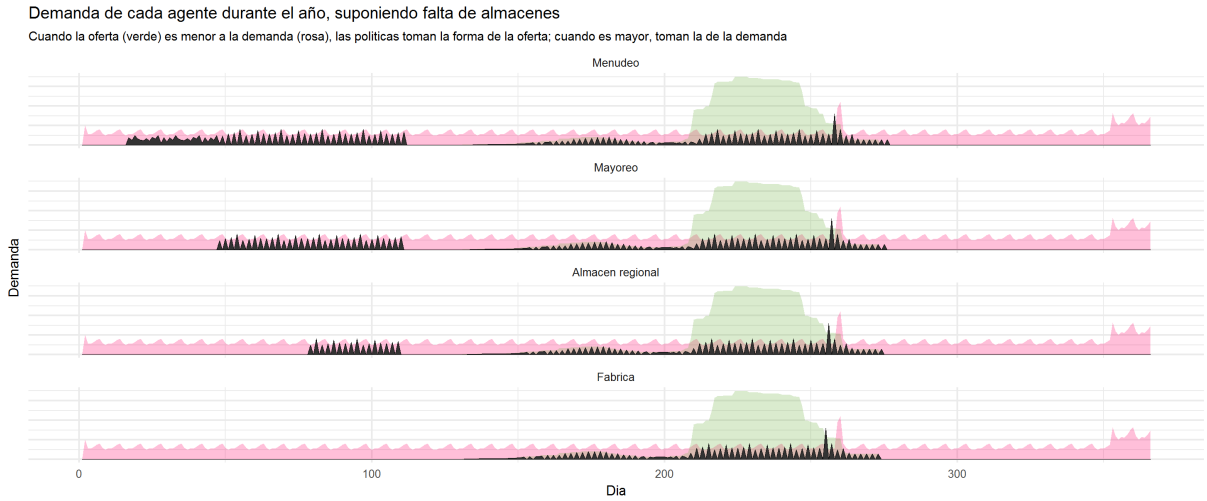


En la figura 5.2 podemos ver el comportamiento de la demanda (el cual fue descrito en capítulos anteriores) en rosa y el de la producción en los campos, ahora con restricción, en verde.

Si todos los agentes tomaran las decisiones relacionadas a la simple máxima "pide hoy lo que esperas vender mañana", entonces el resultado está lejos de ser el óptimo. En este escenario se pueden observar varios efectos:

- Cada agente cuenta con inventario inicial, así que tarda en comenzar a hacer pedidos al agente inmediatamente superior
- Cada agente deja de pedirle al agente inmediatamente superior cuando el segundo ya no tiene inventario. Esto es especialmente notorio para la fábrica: comienza a tener demanda positiva cuando ya hay producción en los campos, cerca del día 150
- Si la oferta es más baja que la demanda, los agentes piden todo lo que haya disponible, pues es preferible cubrir la demanda parcialmente que no cubrirla en lo absoluto
- Los agentes se preparan adecuadamente para el pico de septiembre, con un poco más de anticipación a medida que se encuentran más lejos del consumidor
- Sin embargo, ninguno de los agentes aprende que debería comprar mucho más en el periodo de producción de los campos (en verde) para poder cubrir la demanda de todo el año (en rosa) y maximizar su ganancia

Figura 5.2: Distribución de Cerveza



Como podemos observar, los agentes no se prepararon adecuadamente para cubrir la demanda posterior al tiempo en que los campos tuvieron producción positiva. Esto sucede porque, en este modelo, los agentes ven solamente un periodo hacia el futuro, así que solo se preparan para ese día. Esta complicación vuelve imperante que todos los agentes usen sus almacenes para poder afrontar la demanda, incluso cuando no hay producción. Tal solución se vuelve compleja porque hay muchos parámetros en juego, por mencionar algunos:

- El costo diario de almacenamiento influye directamente en qué tanto tiempo un agente puede mantener la cerveza en inventario para cubrir una venta futura, tal que aún obtenga un margen positivo cuando ésta suceda. Si queremos asegurar que nunca tendrán cerveza guardada durante más de un año, es necesario que se cumpla la restricción:

$$margen_{agente} <= 365 * almacenaje_{agente}$$

- El castigo por orden no cumplida también juega un papel importante en la interacción anterior: si es suficientemente grande, incluso podría resultar en que un margen negativo causado por el costo de almacenamiento es aún preferible a las consecuencias de perder la venta. Si queremos

asegurar que siempre sea preferible buscar una venta, es necesario que se cumpla la restricción:

$$castigo_{agente} >= margen_{agente} + 365 * almacenaje_{agente}$$

- Si la producción total en el año es mayor que la demanda total durante el mismo periodo, no es necesario comprarle toda la cebada a los campos. Paralelamente, si la producción es menor a la demanda, es necesario comprar toda la cebada disponible, pues completar algunas ventas siempre es preferible a no completar ninguna

Sin embargo, algunas de estas condiciones son teóricas y no forzosamente deben cumplirse: es posible que mantener una cerveza en el inventario desde finales de septiembre hasta principios de mayo (cuando vuelve a haber producción) sencillamente no sea redituable.

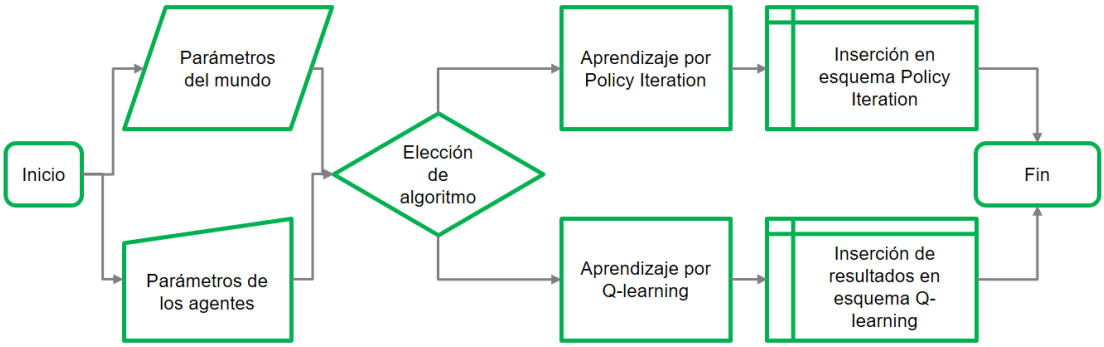
En este trabajo presentaremos un modelo con esta restricción, y lo resolveremos tanto con *policy iteration* como con *Q-learning*. Ambos modelos de aprendizaje reforzado presentados en esta tesis son capaces de capturar todas estas interacciones e incorporarlas al aprendizaje de cada agente.

Capítulo 6

El Producto de Datos

Se creó un pipeline completo que toma los parámetros iniciales, los cuales consisten tanto en condiciones del mundo como en los datos históricos de demanda y oferta; ejecuta el modelo de aprendizaje elegido (ya sea *policy iteration* o *q-learning*; y por último inserta los resultados en una base de datos para hacer posible consulta y análisis posteriores.

Figura 6.1: *Pipeline* del proceso



Al construir la base en la cual se albergan los resultados, se adoptaron las mejores prácticas para creación de bases de datos, esquemas y tablas: siguen una estructura específica comúnmente llamada *tidy* (limpia), en la cual cada variable es una columna, cada observación una fila y cada tipo de unidad observacional es una tabla.

Dado que cada uno de los algoritmos de aprendizaje, *policy iteration* y *q-learning*, tiene requerimentos diferentes y la forma en que presenta resultados es distinta, en la base de datos se construyeron dos esquemas respectivamente, cada uno portando el nombre del algoritmo respectivo.

6.1. Policy Iteration

Es necesario crear tablas que contengan los parámetros necesarios: una para el mundo (tales como la demanda y oferta), otra para los agentes (tales como precios de venta, inventarios iniciales) y una última para los experimentos (los hiperparámetros del algoritmo, tales como ϵ y λ). Por otro lado, dado que este método produce una *policy* en forma de vector de tipo numérico, la manera óptima de almacenar los resultados es en una tabla que contenga como llave el identificador (id) del experimento y el agente, y renglón por resultado.

Como llave de las tablas, se asigna el identificador (id) del experimento como el *hashlib*[1] de la concantenación de ciertos atributos de este: el sello de tiempo en el cual fue ejecutado y las *policies* óptimas de cada uno de los agentes. Esto crea un identificador único y también permite revisar que ningún dato haya sido modificado manualmente, en caso de que tal revisión sea pertinente.

Se obtienen un total de 4 tablas que se relacionan de esta manera[2]:

[Aqui falta el diagrama de las tablas en el esquema Policy Iteration]

6.2. Q-Learning

El resultado de este algoritmo es una tabla que relaciona a cada par de (estado, acción) con un valor de la función Q, así que no es práctico utilizar el mismo formato de tabla de resultados que con *policy iteration*, en el cual un renglón contiene un resultado. Se crea entonces una tabla

[1] Las funciones de *hashing* toman datos de tamaño arbitrario y los mapean a un *hash* de longitud fija.
[2] Representación de los esquemas obtenida directamente del software

por resultado, y una tabla extra que relaciona el identificador (id) del experimento con el nombre respectivo de la tabla. El resto de la estructura del esquema es parecido al de *policy iteration*, pues también existe necesidad de relacionar los parámetros de cada experimento con sus resultados.

Debido a la estructura de la base, el número de tablas es variable, y se relacionan de esta manera:

[Aqui falta el diagrama de las tablas en el esquema Q-learning]

6.3. Especificaciones técnicas

Para el proceso se utilizaron PostgreSQL 10 y Python 3.6.5 en la distribución contenida en Anaconda, así como varios paquetes de Python descritos en el archivo de requerimentos del repositorio de Github. Todos los procesos fueron ejecutados en una computadora Macbook con [] GB de RAM y [] procesadores.

Capítulo 7

Resultados

El problema de llevar inventario de un año a otro se resolvió fácilmente con el método *Q-learning*, pues su misma estructura requiere valores descontados de los estados futuros en cada estado presente. Para lograr el mismo efecto con el método *Policy Iteration*, sería necesario crear artificialmente días de entrenamiento antes y después del periodo de 365 días en el que los agentes están aprendiendo, sin embargo, esto añadiría complejidad computacional.

7.1. El Mundo y los Agentes

A pesar de que nuestro sistema tiene seis roles, en realidad solamente cuatro de ellos toman decisiones:

- El consumidor **obedece** a su necesidad interna de cerveza, mayor en fines de semana y festividades
- La tienda minorista **decide** cuánta cerveza va a comprar a la tienda de mayoreo
- La tienda de mayoreo **decide** cuánta cerveza va a comprar al almacén regional
- El almacén regional **decide** cuánta cerveza va a comprar a la fábrica
- La fábrica **decide** cuánta cerveza va a comprar a los campos
- Los campos **obedecen** a sus ciclos de siembra y cosecha

A pesar de que, estrictamente, en nuestro sistema multiagente todos los roles son agentes, por claridad en la descripción y el flujo del código nos referiremos como agentes solamente a aquellos roles que toman decisiones. Las ecuaciones de los agentes en los extremos "encajan" si suponemos que, para el consumidor, su *policy* óptimo es su demanda normal durante el año, y para los campos, es su producción.

Durante el aprendizaje, todos los agentes siguen el mismo conjunto de reglas (aprender cuánto tienen que comprarle al agente superior en cada día), así que las ecuaciones de ganancia y aprendizaje son las mismas. En este trabajo, usaremos la siguiente construcción del mundo, con cierta notación para los distintos parámetros del mundo:

1. Para cada agente a, el agente superior en la cadena de suministro es $a+1$; el inferior es $a-1$
2. Cada agente tiene un precio de venta p_a, un costo de almacenamiento c_a y un costo por orden no cumplida (*backlog*) b_a
3. En cada día d, cada agente a recibirá del agente superior $a+1$ la cantidad demandada por el primero en el día $d-1$, sujeto a que tal cantidad sea menor o igual a la cantidad que el agente $a+1$ tenía en el almacén
4. En cada una de estas transacciones, cada agente recibe el dinero equivalente, basado en su respectivo precio de venta, e incurre en costos de almacnén y por órdenes no cumplidas

Esto quiere decir que, independientemente del algoritmo de aprendizaje que usemos, cada agente a calcula su ganancia en el día d de la siguiente manera:

$$\begin{aligned} ganancia_{a,d} = &(p_a * ventas_{a,d}) \\ &-(b_a * (demanda_{a-1,d} - ventas_{a,d})) \\ &-(p_{a+1} * ventas_{a+1,d-1}) \\ &-(c_a * inventario_{a,d-1}) \end{aligned}$$

Es decir, cada agente recibe el dinero correspondiente a las ventas que hace al agente inferior, paga el costo castigo relacionado a las órdenes no cumplidas, paga las órdenes hechas al agente

superior y paga el costo de almacenamiento.

Una vez establecidas todas las relaciones entre todos los jugadores y establecidos los parámetros del mundo, hemos obtenido dos formas diferentes de solucionar el juego: la primera con *policy iteration* y la segunda con *Q-learning*.

7.2. Policy Iteration

Las ecuaciones que describen el aprendizaje de un agente son:

1. La ecuación de utilidad (reward) para cada agente a en el día d:

$$R(a,d) = R(a,d) + \gamma * r_{a,d+1} + ... + \gamma^{364} * r_{a,d+364}$$

2. BLA

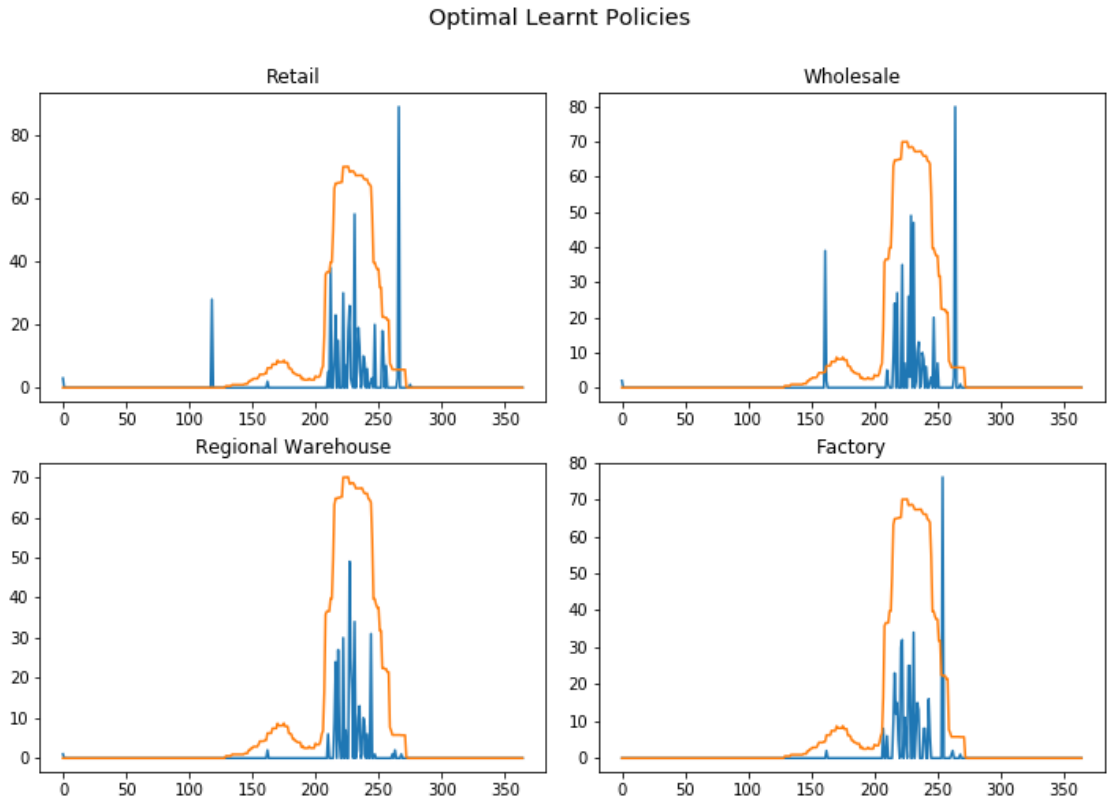
Bajo este algoritmo, los agentes aprenden *policies* que siguen el comportamiento de la oferta, lo cual es el comportamiento esperado dado que en las especificaciones iniciales, se definió manualmente que el la demanda anual no sobrepasara la oferta anual. Este resultado puede observarse en la figura 7.1.

Podemos observar que todos los agentes presentan una tendencia positiva al encontrar mejores *policies* en cada iteración del algoritmo.

7.3. Q-Learning

A diferencia de *policy iteration*, este método encuentra, para cada estado, la acción que acercará al agente lo más posible a la meta. Por supuesto, la *policy* óptima es equivalente con cualquiera de ambos métodos, pero *Q-learning* nos permite más libertad respecto a empezar a jugar el juego a mitad de año, arreglar malas decisiones tomadas por los agentes en periodos anteriores gracias a su adaptabilidad ante cambios estocásticos, etc.

Figura 7.1: Policies óptimas de cada agente bajo *policy iteration*



Las ecuaciones que describen el aprendizaje de un agente son:

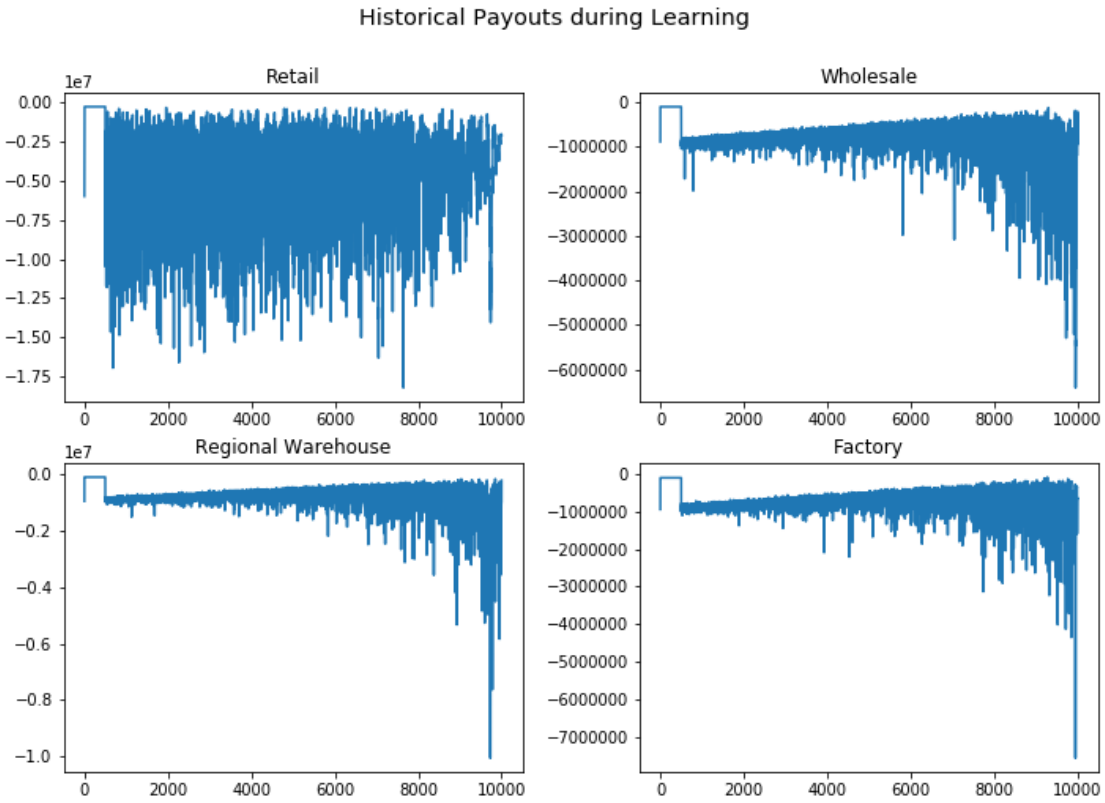
1. La ecuación de utilidad (o recompensa) para cada agente a en el día d:

$$R(a,d) = r(a,d) + \gamma * r_{a,d+1} + ... + \gamma^{364} * r_{a,d+364}$$

   Donde la recompensa del agente a en el día d es la ganancia relacionada a sus respectivas transacciones, como se definió al principio de este capítulo. Es importante notar la importancia de tomar un periodo de un año después del día d sin importar en qué día específico se encuentre el agente: de esta manera, el agente aprenderá si tiene que llegar al día 365 con inventario en almacenes.

2. La función Q para cada agente a dado su estado, el día d con cierto inventario inv:

Figura 7.2: Pagos óptimos de cada agente



$$Q_a(inv_d, compra_d) = r_{a,(d,inv,compra)} + \gamma * \max_{compras} Q_a(inv_d + compra_d, compras_{d+1}$$

Para cada agente, su estado es un vector de longitud 2: en el día d es el inventario que tiene en el almacén, y las acciones que puede tomar en cada estado se representan como un vector de longitud 1: son las diferentes cantidades que puede comprarle al agente superior.

Además, se implementó este algoritmo en la modalidad *greedy*, con ϵ empezando en 1,00 y terminando en 0,05.

Capítulo 8

Conclusiones

Bibliografía

[1] D. Bloembergen y D. Hennes. Fundamentals of multi-agent reinforcement learning. *Tutorial 2: MARL (multi-agent reinforcement learning*, 2013.

[2] L. Busoniu, R. Babuska, y B. De Schutter. Multi-agent reinforcement learning: an overview. *Studies in Computational Intelligence*, 310:183–221, 2010.

[3] S. K. Chaharsooghi, J. Heydari, y S. H. Zegordi. A reinforcement learning model for supply chain ordering management: An application to the beer game. *Decision Support Systems*, 45:949–959, 2008.

[4] Peter Dizikes. The secrets of the system. *MIT News*, 2012.

[5] Junling Hu. Reinforcement learning explained. 2016.

[6] United States Department of Agriculture. Field crops usual planting and harvesting dates. *Agricultural Handbook*, 628:6–7, 2010.

[7] John D. Sterman. Modeling managerial behavior: misperceptions of feedback in a dynamic decision making experiment. *Management Science*, 35:321–339, 1989.

[8] John D. Sterman. *Business Dynamics*. Irwin/McGraw-Hill, 2000.

[9] F. Strozzi, J. Bosch, y J.M. Zaldivar. Beer game order policy optimization under changing customer demand. *Decision Support Systems*, 42:2153–2163, 2007.

[10] R Sutton y A. Barto. *Reinforcement learning: an introduction*. The MIT Press, 1998.

[11] Mohammad Zarandi, Mohammad Hassan Anssari, Milad Avazbeigi, y Ali Mohaghar. A multi-agent model for supply chain ordering management: An application to the beer game. *Supply Chain Management*, págs. 433–442, 2011.