Slide 0

Buenos días, soy Dani y os voy a presentar mi tésis de máster dirigida por Toni Lozano. El trabajo ha estado motivado por un proyecto sobre Product Delvuery que se realizó en Grupo AIA, la empresa dónde he estado haciendo las prácticas, y dónde usaron técnicas clásicas de optimización como seria el Símplex. Nuestro objetivo ha estado resolver ese problema con Machine Learning, usando técnicas de Reinforcement Learning y redes neuronales.

1. Problem

Slide 1

Supongmos que somos propietarios de una cadena de tiendas especializadas en cierto producto; y que pagamos a una empresa de transporte para que redistribuya nuestro producto des de un cargadero hasta las tiendas.

Nuestro objetivo es minimizar el coste total a pagar a la empresa de transporte, asegurando que siempre hay producto en stock en todas las tiendas.

Slide 2

Por ahora consideramos n tiendas y k camiones que pueden ir a una única tienda al día, o bien quedarse en el cargadero.

Una restricción fuerte que consideramos es que los camiones deben salir del cargadero llenos y vuelver al final del día completamente vacíos.

Por último consideramos que el coste a pagar a la empresa de transporte viene dado por cierta función de coste J_{costs}.

Slide 3

Además del coste económico de transporte, consideramos un criterio cualitativo para decir como de " bueno" es el nivel de stock de una tienda.

Imaginemos que nuestra tienda guarda todo su producto en un contenedor cilíndrico.

Básicamente distinguiriamos:

- la máxima y mínima capacidad de las tiendas, que significarían una restricción fuerte,
- El nivel mínimo de stock, que seria el consumo previsto para las próximas 12 horas,
- El nivel de "peligro" que seria el consumo previsto para las próximas 36 horas,
- y finalmente el nivel máximo de stock, que sería el máximo nivel de stock deseado.

De esta forma nos queda definida una zona óptima donde desaríamos mantener el nivel de stock de las tiendas.

La idea será definir un coste a partir de este criterio para dar más peso a que las tiendas tengan un nivel de stock en la zona verde y nunca tiendan a ir a los niveles más extremos.

Slide 4

Por lo tanto, lo que queremos resolver es un problema de optimización donde tendremos un *trade-off* entre minimizar los costes a pagar a la compañía de transporte y el bienestar de las tiendas.

2. Reinforcement Learning (RL) concepts

Slide 5

En el contexto del Reinforcement Learning se considera que tenemos un agente (que sería el software o algoritmo a entrenar), que vive en un entorno que puede encontrarse en varios estados posibles.

En el caso del tres en ralla, el entorno seria el tablero y los posibles estados serian las posibles configuraciones de fichas.

Luego tenemos el concepto de las acciones que el agente puede tomar, y en el caso del tres en ralla sería en qué posición del tablero tirar la ficha.

Slide 6

El modelo matemático que se usa para modelizar un sistema en Reinforcement Learning són los Markov Decision Processes (MDP).

Básicamente son Cadenas de Markov, donde hay estados y transiciones entre estados con cierta probabilidad, pero ahora además se añaden las acciones, que podemos considerar como estados intermedios.

La idea es la siguiente: supongamos por ejemplo que estamos en el estado s_0. En esta situación hay dos posibles acciones a tomar a_0 y a_1. Si tomamos por ejemplo la acción a_0, con probabilidad un medio volveremos al estado s_0 y con probabilidad un medio el sistema hará una transición hacia el estado s_2.

Adicionalmente se introduce el concepto de *reward*, que seria como una recompensa o penalización que recibe el agente según si la acción tomada es buena o mala.

Formalmente una MDP viene dada por un conjunto de estados, un conjunto de acciones, una función de probabilidad de transición y una función de rewards.

Slide 7

Una vez definido el sistema partir de una MDP, estariamos interesados en como controlar el sistema mediante el agente que queremos entrenar.

Para ello se introduce el concepto de policy, que en el caso más simple sería una función que dado un estado nos diga qué acción tomar. Y en general tendríamos una policy estocástica, que nos daría la probabilidad de tomar cada acción dado un estado.

El objectivo final seria encontrar la policy óptima que maximize los rewards obtenidos por el agente.

Slide 8

Una vez tenemos una policy como la usamos para controlar el sistema??

La idea es empezar por un estado inicial, aplicar la policy para determinar la acción a tomar y entonces el sistema hará una transición a un nuevo estado obteniendo cierta cantidad de reward. Iterando el proceso obtendriamos una sequencia de estados, acciones y rewards que si la consideramos finita de longitud τ diriamos que tenemos un episodio de longitud τ.

3. Reinforcement Learning Model for product delivery

Slide 9

Ahora como planteamos nuestro problema de product delivery en el contexto de Reinforcement Learning?

En nuestro caso el entorno seria el mundo dónde tenemos n tiendas, el cargadero y k camiones.

Para las simulaciones consideraremos episodios de 30 días cada uno, así que estariamos resolviendo el problema de subministrar a las tiendas "a 30 días vista".

Los estados vendrian dados por los niveles de stock disponibles en cada tienda,

y las acciones vendrían definidas por un número entero para cada camión que nos diría a qué tienda debe ir, o si debe quedarse en el cargadero.

Slide 11

Como función de rewards consideramos básicamente tres términos. El primero será para tener en cuenta los costes económicos que tenemos que pagar a la empresa de transporte.

El segundo sería un coste por niveles de stock para modelizar el bienestar de las tiendas.

Y por último añadiríamos términos que ayudarían a penalizar acciones no permitidas. Por ejemplo, acciones que llevaran camiones a una tienda demasiado llena como para que el cargamento del camión quepa en ella.

Slide 12

Para el coste de niveles consideramos una función de esta forma para cada tienda:

En la zona optima consideramos rewards positivos y a medida que nos alejamos de esta zona penalizamos negativamente. La idea es penalizar de forma exponencial en las zonas extremas para evitar que las tiendas se vacíen o se llenen en exceso.

Slide 13 (Summary)

Queremos encontrar una policy que maximize los rewards. Tenemos definidos los estados, las acciones y la función de rewards, pero no conocemos la función de transición T.

Por ello, necesitamos algoritmos que no necesiten del conocimiento previo de T. Nosotros nos hemos centrado en el llamado Q-learning.

4. Applying RL: Q-learning algorithm

Slide 14

El Q-learning se basa en los llamados Q-values que básicamente son un cuantificador de como de "bueno" es tomar una cierta acción partiendo de un estado en concreto.

Es la esperanza de los discounted rewards. Para fijar ideas, tomad $\gamma = 1$ de forma que este término seria la suma de todos los rewards a lo largo del episodio y los Q-values serian el valor esperado de los rewards totales.

Slide 15

A partir de los Q values podemos definir lo que seria una policy optima π^*, que seria aquella policy que maximiza el Q value para todo par de estados y acciones.

En particular, si conociésemos los Q-values óptimos, y definimos ésta policy (para cada estado coger la acción con máximo Q-value), tendremos una policy óptima.

Slide 16

Para aprender los Q-values óptimos se simulan episodios usando una policy que nos ayude a explorar el espacio de estados y acciones, y entonces se aplica un algoritmo iterativo llamado Q-learning que va actualizando los Q-values hasta converger a los valores óptimos (Teorema, si preguntan).

4.1. Simulations

Slide 17

Para las simulaciones de Q-learning consideramos 5 tiendas y dos camiones.

Además, discretizamos los estados en 4 subintervalos. Es decir, dividimos cada tienda en 4 partes iguales en ve de tener un rango infinito de valores posibles para el stock.

Cada simulación consta de 500.000 episodios de 30 steps cada uno.

Slide 18

Consideraremos 4 tipos de simulaciones. Por un lado deterministas, si consideramos que las tiendas consumen una misma cantidad de producto al final de cada día; o estocáticas, si introducimos ruido al producto que se consume cada día. Luego por un lado consideraremos el caso en que no haya costes de transporte y por otro sí (cuando μ_1 sea zero y cuando no).

Slide 19,20,21,22

Para todas las simulaciones vamos a utilizar el siguiente tipo de gráfico. Tendremos un subplot para cada tienda mostrando el nivel de stock a lo largo del episodio simulado usando la policy aprendida con Q-learning. Las línias discontínuas nos indicarian los niveles de stock de acuerdo con la figura de la derecha, en azul tendriamos el valor promedio de stock durante el episodio y en verde claro el nivel de stock que nos daria el máximo reward posible.

1) Slide 19

Comenzamos comparando el caso determinista sin y con constes de transporte.

Sin costes de transporte vemos que se envian 43 camiones en total, y el nivel de las tiendas se mantiene prácticamente siempre en la zona óptima (por encima de la línea verde).

2.1) Slide 20 Si ahora damos un poco de peso al coste de transporte, vemos como las tiendas siguen manteniéndose en la zona verde, pero ahora se envían 3 camiones menos y el coste de transporte se ha reducido en unas 2000 unidades.

Si comparamos con la simulación anterior, vemos que la diferencia está en las tiendas 1 y 3, dónde vemos que el nivel de stock promedio se ha reducido a cambio de reducir también los costes de transporte.

2.2) Slide 21 Si damos más peso a los costes, aún vemos que las tiendas se mantienen bastante bien en la zona verde, pero si nos fijamos comparando con la simulacion anterior, el nivel medio de las tiendas (línea azul) baja un poco más en todas las tiendas.

A cambio, el coste de transporte se ha reducido unas 1000 unidades más, habiéndose enviado 3 camiones menos con respecto al caso anterior.

2.3) Slide 22

Para acabar os quería mostrar que si vamos dando más y más peso a los costes de transporte, llega un punto en que se da más prioridad al precio que a mantener las tiendas vivas, de forma que algunas de ellas se vacían.

De aquí el trade-off del que hablábamos al principio.

Slide 23,24

Consideramos ahora simulaciones donde las tiendas consumen como en el caso determinista pero con cierto porcentage de ruido.

3.1) Sin costes de transporte las tiendas se mantienen casi siempre en la zona óptima, se envian 42 camiones,

4.1) y considerando costes de transporte vemos que se envía 1 camión menos, se sacrifica un poco el nivel de las tiendas, pero a cambio se reduce el coste de transporte.

Con esto vemos que Q-learning es capaz de aprender también en entornos estocásticos.

Slide 25

Con esto concluimos que Q-learning seria efectivo para un número de tiendas y camiones pequeño, como serían 5 tiendas y 2 camiones, dónde el espacio de estados y acciones tendria unos 36800

elementos, mientras que si queremos 20 tiendas y 3 camiones este número se va al orden de 10^{16}, lo qual es inviable para Q-learning.

Además esto seria si discretizamos las tiendas en solo 4 partes, lo que nos estaria haciendo perder mucha información sobre el sistema.

5. Deep Reinforcement Learning

Slide 26

El siguiente paso seria hacer el problema escalable a un número mayor de tiendas y camiones.

Para ello recurrimos a las redes neuronaes aplicadas a reinforcement learning, en nuestro caso usando el Policy Gradient algorithm.

La idea es usar una red neuronal como policy parametrizada con ciertos parametros θ, que serían los pesos y los biases de la red, y actualizar estos pesos para optimizar nuestra policy.

Slide 27

La arquitectura de red que consideramos es la siguiente. Por un lado como inputs tomariamos los niveles de stock de cada tienda, que seria el estado del sistema en un momento dado, luego 2 hidden layers y finalmente un softmax layer que como output que nos daria las probabilidades de tomar cada una de las posibles acciones. Por lo tanto tenemos $(n+1)^k$ neuronas en el output, una para cada acción posible.

5.1. Monte Carlo Policy Gradient

Slide 28

Una vez tenemos construida la red neuronal necesitamos saber como entrenarla. Para ello consideramos ésta función de coste J_θ a maximizar.

Y consideramos el Policy Gradient Theorem que nos da una expresión exacta para calcular el gradiente de ésta función de coste a partir de ésta expresión.

Slide 29

A la práctica lo que se hace es simular un conjunto de N episodios, y estimar numéricamente esa esperanza haciendo un promedio.

Finalmente, a partir del gradiente y algún método de optimización, como seria el método de Gradient Ascent, podemos actualizar los parámetros de la red.

5.2. Simulations

Con ésto pasamos ya a considarar simulaciones análoproducto a las que hemos visto para el Q-learning pero ahora usando una red neuronal como policy.

1) Slide 30

2.1) Slide 31

Para el caso determinista, si comparamos el caso sin costes y con costes, vemos que se envian el mismo número de camiones (41), pero en el segundo casos la red es capaz de enviar los camiones de forma que los costes de transporte se reduzcan. A cambio se sacrifica un poco el bienestar de las tiendas.

3) Slide 32

4) Slide 33

Para el caso estocástico, sin costes de transporte se envian 49 camiones y el bienestar de las tiendas es muy bueno, y con contribución por costes de transporte vemos que se envian 9 camiones menos, el coste total se reduce, pero a cambio el bienestar de las tiendas también disminuye un poco.

5) Slide 34

Finalmente podemos mostrar que el nuevo método usando redes neuronales, escala mejor que Q-learning. En esta simulación, estocástica y sin costes de transporte consideramos 12 tiendas y 3 camiones. Vemos que el nivel de las tiendas se mantiene de forma razonable,

6) Slide 35

y si pasamos a poner un poco de contribución por costes de transporte, vemos que el bienestar de las tiendas se reduce un poco pero se envian 2 camiones menos y los costes totales se reducen.

6. Conclusions and Future work

Ya para acabar pasamos a las conclusiones.

Slide 36

Hemos visto que Q-learning escala muy mal con el número de tiendas y camiones. En concreto si d es el número de partes en que discretizamos las tiendas, la dimensión del espacio de estados y acciones escala exponencialmente con k y n de ésta forma (señalar expresión).

Contrariamente, con el método de Policy Gradient y redes neuronales, la dimensionalidad del espacio de estados es irrelevante y lo que limita en este caso la escalabilidad és el número total de acciones posibles, ya que seria igual al número de neuronas en el output layer y entrenar una red con un layer del orden de 10^4 o más neuronas es inviable con un tiempo de entrenamiento razonable.

Slide 37

Es por esto que no hemos considerado más de 3 camiones y 12 tiendas.

Con n igual a 12 y k igual a 3 tendriamos unas 2200 neuronas en el outputlayer, pero si consideramos un camión más, el número de neuronas ya se va a más de 28 mil.

Para aliviar el problema de escalabilidad debido al número de neuronas en el output, que serian $(n+1)^k$, proponemos una arquitectura de red neuronal que pasaria de escalar exponencialmente a escalar linealmente con $(n+1) \cdot k$ neuronas en el output.

Slide 38

Básicamente tendríamos la misma arquitectura, con dos hidden layers, pero ahora en vez de tener un único softmax layer con tantas neuronas como acciones posibles, tendríamos un softmax layer para cada camión, con $(n+1)$ neuronas cada uno.

De este modo cada softmax layer nos daria las probabilidades de ir a cada tienda o quedarse en el cargadero para cada uno de los camiones.

La difultat estará en diseñar un algoritmo que nos permita entrenar ésta red para que aprenda. Y si lograsemos esto probablemente podriamos publicar algun paper .

pongo ésto del paper? yo creo que estaría bien

Slide 39

Ya para acabar, solo decir que todo el material de la tésis lo hemos puesto en este repositirio github y en este otro tenemos el entorno de Open AI que se ha programado para resolver nuestro problema de product delivery.