1. Problem

Slide 1

Nuestro cliente es propietario de una cadena de tiendas especializadas en cierto producto; en nuestro caso gas natural, que paga a una empresa de transporte para que redistribuya su producto des de el puerto de Barcelona a las tiendas.

Nuestro objetivo es minimizar el coste que Gas Natural tiene que pagar a la empresa de transporte, asegurando que siempre hay gas disponible en todas las plantas, entre otras restricciones.

Slide 2

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

Una restricción fuerte que consideramos es que los camiones salen del cargadero llenos y vuelven al final del día completamente vacíos.

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

Slide 3

Además del coste económico de transporte y demás, Gas Natural tiene un criterio para decir como de " bueno" es el nivel de stock de una tienda.

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.

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 bajos.

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 nivel de stock de las tiendas.

2. Reinforcement Learning (RL) concepts

Slide 5

Básicamente podemos entender que Reinforcement Learning són un conjunto de técnicas para aprendizaje automatizado que se ha usado des de hace mucho tiempo en Inteligencia Artificial. Por ejemplo se usa para entrenar a bots en videojuegos, robots reales como podrían ser vehículos automatizados, en recomendadores de canciones como en You Tube, marketing, etc.

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, imaginemos que estamos jugando contra la máquina, que seria el agente. El entorno seria el tablero y los posibles estados serian las posibles configuraciones de fichas.

Luego tenemos el concepto 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 las Markov Decision Processes.

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 systema 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.

Slide 7

Para formalizar las Markov Decision Processes necesitamos por una función de transición T, que daría la probabilidad de transicionar de un estado s a un estado s' después de haber tomado la acción a.

Slide 8

Y una función de reward R, que sería el equivalente a una función de coste en problemas clásicos de optimización. Valores positivos serían para recompensar buenas acciones y valores negativos seria para penalizar las malas.

Slide 9

Con esto se define una Markov Decision Process como una cuádrupla dada por un conjunto de estados, un conjunto de acciones, una función de transición y una función de rewards.

Slide 10

Una vez definido el systema modelo a 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 sería una función que mapea estados a acciones. Y el objetivo seria encontrar la policy óptima que maximizase los rewards.

La idea es que entrenar o hacer aprender al agente es equivalente a mejorar la policy.

Slide 11

Una vez tenemos una policy como la usamos?

La idea es empezar por un estado inicial s_0, aplicar la policy para determinar la acción a_0 a tomar y entonces el sistema hará una transición a un nuevo estado s_1. 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.

Slide 12

Resumiendo, para aplicar Reinforcement Learning a un problema, necesitamos primero de todo una Markov Decision Process,

y el objectivo es encontrar una policy que maximize los rewards. En concreto se quedría maximizar el "discounted sum of rewards" que vendría dado por esta expresión.

La idea de poner el factor gamma menor que 1 es para dar menor importancia a las recompensas futuras y dar más peso al presente y al futuro cercano.

3. Reinforcement Learning Model for product delivery

Slide 13

Para el problema de Gas Natural, el entorno seria el mundo dónde tenemos n plantaciones, 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 plantas "a 30 días vista".

Los estados los definimos como un vector con tantas componenentes como plantas y con valores igual al stock de gas disponible en un momento dado.

Las acciones serían decidir dónde llevar cada camión (a qué planta o si se queda en el cargadero).

Slide 14

En esta slide sólo queria remarcar que el número de acciones posible crece muy rapidamente con el número de camiones y de plantas.

Con 5 plantas y 2 camiones tendriamos 36 acciones posibles, pero si nos vamos a 30 plantas y 5 camiones ya llegamos al millón de acciones.

Por otro lado los posibles estados del systema serian infinitos, ya que el nivel de stock de cada planta en principio puede ser un número real entre 0 y la capacidad máxima de la planta. Por ello lo que haremos es discretizar los niveles de las plantas.

Slide 15

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

El segundo sería para dar mayor o menor importancia a los niveles de stock de cada planta.

Y por último añadiríamos términos que ayudarían a penalizar por ejemplo acciones que llevaran camiones a una planta demasiado llena como para que el cargamento del camión quepa en ella.

Slide 16

Como función de coste consideramos que sea proporcional a la distancia viajada por los camiones y a la cantidad total de producto que dejan en las plantas.

Slide 17

Para el término de niveles de stock tomamos una suma de términos, uno para cada planta y que será función del porcentaje de gas que le queda a esa planta.

Slide 18

Explicar la gráfica:

En principio lo ideal seria que las plantas tuviesen su nivel de stock en la zona verde por lo que damos reward positivo en esa zona con un cierto valor máximo en un valor intermedio.

Para la zona entre el “Danger level” y el minimum level consideramos recta decreciente con valores negativos.

Y finalmente, para las zonas por debajo del minimum level y el maximum level consideramos una función exponencial que tome valores cada vez más negativos. La idea es que teóricamente tomaría el valor infinito en el nivel de capacidad máxima y mínima. Pero por razones numéricas truncamos en un valor fijado P2.

4. Applying RL: Q-learning algorithm

Slide 19

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.

Slide 20

El Q-learning se basa en lo que llamamos 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.

Slide 21

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 22

Para aprender los Q-values óptimos se hace a base de generar episodios usando una policy que nos ayude a explorar el espacio de estados y acciones. La forma más exploratoria sería partir de un estado inicial e ir tomando acciones de forma totalmente aleatoria. El otro extremo sería usar la policy de la diapositiva anterior a partir de los Q-values aprendidos hasta el momento.

Slide 23

Normalmente se usa una policy intermedia que básicamente consta en escoger una acción de forma totalmente aleatoria con cierta probabilidad ε o seguir la policy basada en los Q-values con probabilidad $1 - \varepsilon$.

Slide 24

Para aprender los Q-values óptimos, se simulan varios episodios.

Vamos a pensarlo para el problema de las plantas y los camiones:

- Al principio de cada episodio inicializariamos el nivel de stock de cada planta, lo que nos daria el estado inicial.
- Luego, en cada uno de los 30 dias que dura el episodio, escogeriamos la acción a llevar a cabo, que en nuestro caso seria decir dónde va cada camión.

- Entonces se aplicaria dicha acción: para nosotros seria llenar las plantas donde han ido los camiones y a la vez reducir el stock de cada una de ellas para simular el consumo diario de estas.
- Una vez obtenido el nuevo estado, y sabiendo el estado y accion anteriores, se calcula el reward obtenido.
- Con esto, por último lo que se hace es actualizar el Q-value correspondiente al estado anterior con esta fórmula (el reward actual + el Q value máximo).

4.1. Simulations

Slide 25

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

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

Cada simulación consta de 100.000 episodios de 30 steps cada uno, que serian unas 7/8 horas de training.

Slide 26

Consideraremos 4 tipos de simulaciones. Por un lado deterministas, si consideramos que las plantas consumen una misma cantidad de gas al final de cada día; o estocático, si introducimos ruido al gas que se consume cada día. Luego por un lado consideraremos el caso en que no haya costes de transporte y por otro sí.

Para tener una idea de lo que ha aprendido nuestro algoritmo en todas las simulaciones que hemos hecho os enseño un giff para ver como se comporta el sistema al ser controlado con Q-learning.

(PONER GIFF)

Slide 27,28,29,30

1) Sim 16 Para la primera simulacion considramos que las plantas consumen la misma cantidad de gas y sin considerar los costes de transporte. Por lo tanto solo estamos teniendo en cuenta el coste de niveles, por lo que esperariamos que las plantas se mantuvieran en la zona verde (que seria de la línea verde oscuro a la línia naranja superior).

2.1) Sim 17 Si ahora damos un poco de peso al coste de transporte, vemos como las plantas siguen manteniendose en la zona verde, pero ahora se envia un camion menos y el coste de transporte se ha reducido en 9 unidades.

Si comparamos con la simulación anterior, vemos que la única diferencia entre las dos es que en ve de enviar dos camiones pequeños de 70 a la planta 1, se envia uno grande de 130 (pizarra para escribir porqué esto implica que es más barato?)

2.2) Sim 18 Podemos ahora dar un poco más de peso al transporte. Aún parece que las plantas se mantienen bastante bien en la zona verde, pero si nos fijamos comparando con la simulacion anterior, el nivel medio de las plantas (línea azul) baja un poco en todas las plantas.

Lo bueno es que ahora el coste de transporte se ha reducido hasta 2777 sin tener que haber reducido el número de camiones.

Ver porqué: (Libreta Dani)

2.3) Sim 22

Para acabar con las simulaciones deterministas os queria enseñar que si vamos dando más y más peso al transporte llega un punto en que da más prioridad el coste que a mantener las plantas vivas, de forma que algunas de ellas se vacían.

Slide 31,32,33,34

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

3.1) Sim 14 Vamos a comparar los casos con costes de transporte y sin costes de transporte.

Lo primero que vemos es que aún teniendo bastante stocasticidad, el algoritmo es capaz de mantener las plantas en la zona verde.

Sin costes de transporte se envian 40 camiones y el precio seria de 2868.

4.1) Sim 19

Si ahora consideramos costes de transporte, se siguen enviando 40 camiones, los niveles siguen en torno a la zona verde y ahora además el precio se ha reducido de 2868 a 2820.

Ahora lo interesante es ver como lo ha hecho el agente para reducir el coste de transporte sin reducir el número de camiones enviados.

Si contamos el total de camiones pequeños y grandes enviados en cada caso, vemos que sin costes de transporte se envian 8 pequeños y 19 grandes, mientras que con costes de transporte se envian 9 pequeños y 18 grandes.

Como la diferencia de envio de camiones está entre las plantas 3 y 4, y la distáncia que hay del cargadero a ellas es la misma, efectivamente és más barato enviar 1 camión pequeño en vez de uno grande.

5. Learning a policy with Deep Neural Networks

Para aprender redes neuronales con tensorflow he fabricado un ejemplo sencillo de classificación adaptado al problema de las plantas y los camiones.

Slide 35

La idea ha sido considerar 3 plantas y un camión, de forma que los estados serian simplemente vectores de 3 componentes, cada uno siendo el nivel de gas que queda en cada planta. Y las acciones serían o bien ir a una de las tres plantas, o bien quedarse en el cargadero.

Slide 36

El objetivo es hacer aprender a una red neuronal una policy senzilla, que lleve el camión a la planta con menos gas si puede, es decir, si la carga que lleva el camión cabe completamente en la planta a la cual pretende ir.

Así, si generamos aleatoriamente un training set de estados y como target consideramos aplicar esta policy a ese dataset; estaremos en el framework de un problema de clasificación.

Slide 37

Para abordar el problema con redes neuronales consideramos una red con la siguiente estructura: 3 inputs, uno para el nivel de stock de cada planta, y 4 outputs, uno para cada acción posible. El único detalle es que en el output añadimos una activación softmax para que los outputs sumen uno. Y la acción a llevar a cabo seria la que tubiese un output mayor.

Slide 38

Explicar rápido idea o pasar.

5.1. Simulations

Para las simulaciones hemos considerado diferentes números de neuronas en cada layer y entrenado la red durante 3 minutos (1000 epocas).

Slide 39

Con 100 neuronas en el primer layer y 50 en el segundo vemos como en unas 10 o 20 epocas la accuracy ya supera el 95 % y más o menos oscila entorno al 98 %.

Slide 40

Con 20 neuronas en el primero y 10 en el segundo obtenemos prácticamente los mismos resultados en accuracy, solo que necesitamos un poco más de 100 épocas para ...

Slide 41

Con 4 y 4 vemos que los resultados no son tan buenos, pero aún así la red és capaz de aprender la policy estabilizando en una accuracy en torno al 95 %.

Slide 42

Para acabar, en estos 8 gráficos podemos ver diferentes test runs para provar la policy aprendida: señalar algun gráfico o poner el GIFF mejor.