INFORME DE PRACTICAS DE FSI – DAVID SUÁREZ SUÁREZ

Introducción

En primer lugar, a modo de introducción decir que en este informe se reflejarán los resultados (texto, tablas, gráficas, etc.) fruto de la ejecución de las tres prácticas. Además, se incluirán en la medida de lo posible breves explicaciones a los mismos a fin de contrastarlas con los conceptos teóricos impartidos en la asignatura.

Además de esto, se dejarán algunas aclaraciones del código de cada práctica para reflejar la estrategia o pasos seguidos a la hora de elaborarlo. Lo que se quiere decir es que no entraremos en detalles concretos del código, pero si comentaremos el "enfoque" que se tomó para afrontar la elaboración del mismo. Naturalmente para más detalles el código está disponible (y comentado incluso en exceso para reflejar todos los detalles que aquí no se recogen).

Practica 1 – Estrategias de búsqueda

Breve explicación del trabajo realizado

En esta práctica se realizaron ampliaciones del código para añadir tanto la "búsqueda por ramificación y acotación" como la de "búsqueda por ramificación y acotación con subestimación". En nuestro caso, lo único que se tuvo que hacer fue añadir las líneas necesarias en el "run.py" para ejecutar la búsqueda relativa a estos dos últimos tipos de búsqueda y añadir las estructuras necesarias en el resto del código base.

Estas estructuras consistieron básicamente en implementar un nuevo tipo de cola que lo único que hace diferente de una cola estándar es ordenar los elementos que contiene tras añadir nuevos elementos (atendiendo a un criterio determinado). En nuestro caso, ordenará atendiendo al coste para cada nodo (búsqueda por ramificación y acotación) o bien atendiendo al coste más la heurística para cada nodo (búsqueda por ramificación y acotación con subestimación). De este modo, se irán explorando (expandiendo) los nodos con menor coste (o coste más heurística), pues se colocan de manera que sean los primeros en ser extraídos de la cola.

Para ejecutar este tipo de búsqueda tan solo debemos de invocar a la función que realiza la búsqueda en árbol (pasándole el tipo de lista previamente comentada).

Resultados

A continuación se muestra una tabla con algunos ejemplos de búsquedas realizadas, mostrando el número de nodos expandidos por cada tipo de búsqueda: anchura, profundidad, ramificación y acotación (RA) y ramificación y acotación con subestimación (RAS).

Notar que los resultados son muy diversos según el tipo de búsqueda realizada y el contenido de la búsqueda. Asimismo, aclarar que la heurística es muy determinante en el proceso de búsqueda, ya que al tratarse de una simple estimación puede alejarte o acercarte más al nodo objetivo (esto depende de si la heurística es buena o mala, que depende de si subestima lo justo o subestima demasiado; también puede ocurrir que sobreestime, lo cual no es bueno en una heurística ya que descarta soluciones probablemente buenas suponiendo que son malas).

	ANCHURA	PROFUNDIDAD	RA	RAS
M → F	13	7	102	28
T → F	10	16	42	8
R → D	8	2	13	4
D → O	13	6	68	14
B → A	13	5	64	4
A → N	19	14	2468	270

Practica 2 – Redes Neuronales

Breve explicación del trabajo realizado

En esta práctica lo que se trato en primer lugar fue el caso de la planta del Iris. Ya venía la mayor parte del código y tras probarlo y comprenderlo, se realizó la modificación pertinente que solo consistió en separar el conjunto de la forma solicitada. De esta forma, tenemos un 70% de los datos destinados a que la red aprenda (conjunto de entrenamiento), otro 15% destinado a contrastar/validar lo que la red aprende en cada época/entrenamiento (conjunto de validación) y otro conjunto destinado a probar si una vez completado el entrenamiento, la red es capaz de actuar con una buena tasa de acierto frente a datos que nunca ha tratado directa o indirectamente (conjunto test). En cuanto a la estructura de la red se dejó intacta (no se añadieron capas ocultas ni neuronas).

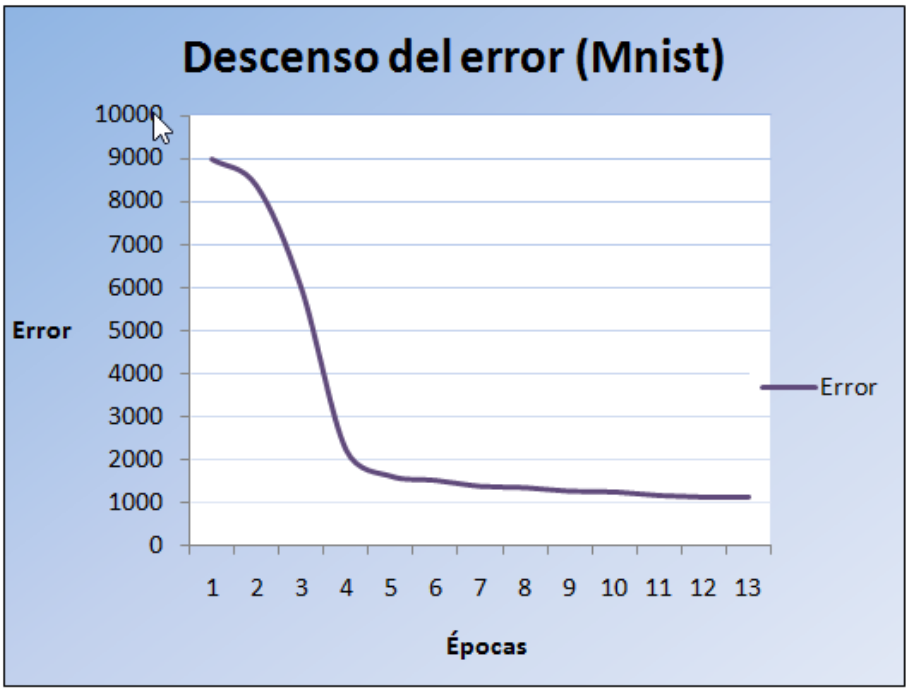
En el caso del Mnist, el procedimiento es simplemente el mismo pero realizando algunas modificaciones en la red neuronal. Los cambios realizados afectan al número de neuronas de la red en cada capa (entrada, salida y oculta) y al número de capas de la red (se añadieron capas ocultas). Todas estas modificaciones a fin de garantizar el funcionamiento de la red y mejorar su eficiencia (ya que este problema es bastante más complejo que el de la planta del Iris). Asimismo, fue necesario cambiar el número de entradas a la red ya que ahora tenemos como entrada todos los pixeles de cada una de las imágenes del conjunto Mnist (para el problema de la planta del Iris eran el alto y ancho de dos partes de la planta).

Aclarar que en el primer problema (planta del Iris) se entrenó a la red un número de épocas arbitrario, concreto y exacto solo para ver la evolución del error (que iba decreciendo hasta cierto punto). Obviamente esto no es lo recomendable ya que no garantizamos ni que se maximice el aprendizaje de la red ni que se realicen las épocas mínimas para ello. Por otro lado, en el segundo (Mnist) ya se realizó de la manera recomendable, puesto que se realiza el número de épocas necesario hasta encontrar el mínimo relativo de la función de error (cuando el error crece en lugar de disminuir). Como ya dijimos, esto es lo conveniente ya que te asegura disminuir el error lo máximo posible en el número mínimo de épocas.

Otra aclaración es que la evolución del error hasta que este alcanza el valor más pequeño posible depende de los valores iniciales de los pesos de cada entrada a cada neurona en la red. Dichos valores recordemos que comienzan en valores totalmente aleatorios y por tanto en cada ejecución podemos obtener resultados diferentes en la minimización de nuestro error (desde disminuirlo al mínimo absoluto de la función hasta estancarse en el mayor mínimo relativo).

Resultados

Época	Error
1	8992,25
2	8371,35
3	5993,28
4	2226,53
5	1614,77
6	1516,33
7	1378,87
8	1346,03
9	1264,09
10	1245,28
11	1163,41
12	1123,78
13	1128,17



Naturalmente, estos son los resultados de una ejecución y no son iguales en cada una de las ejecuciones (como ya se comento antes). No obstante, si existen cosas que se pueden sacar en claro: el descenso en un momento u otro de la función de error hasta un mínimo relativo y los resultados del conjunto test. Los resultados del test no son idénticos pero casi todos rondan en torno al 7% y 10% (tasa de fallos). En la mayoría de las ejecuciones el error desciende hasta 1000 aproximadamente.

Practica 3 – Aprendizaje por refuerzo

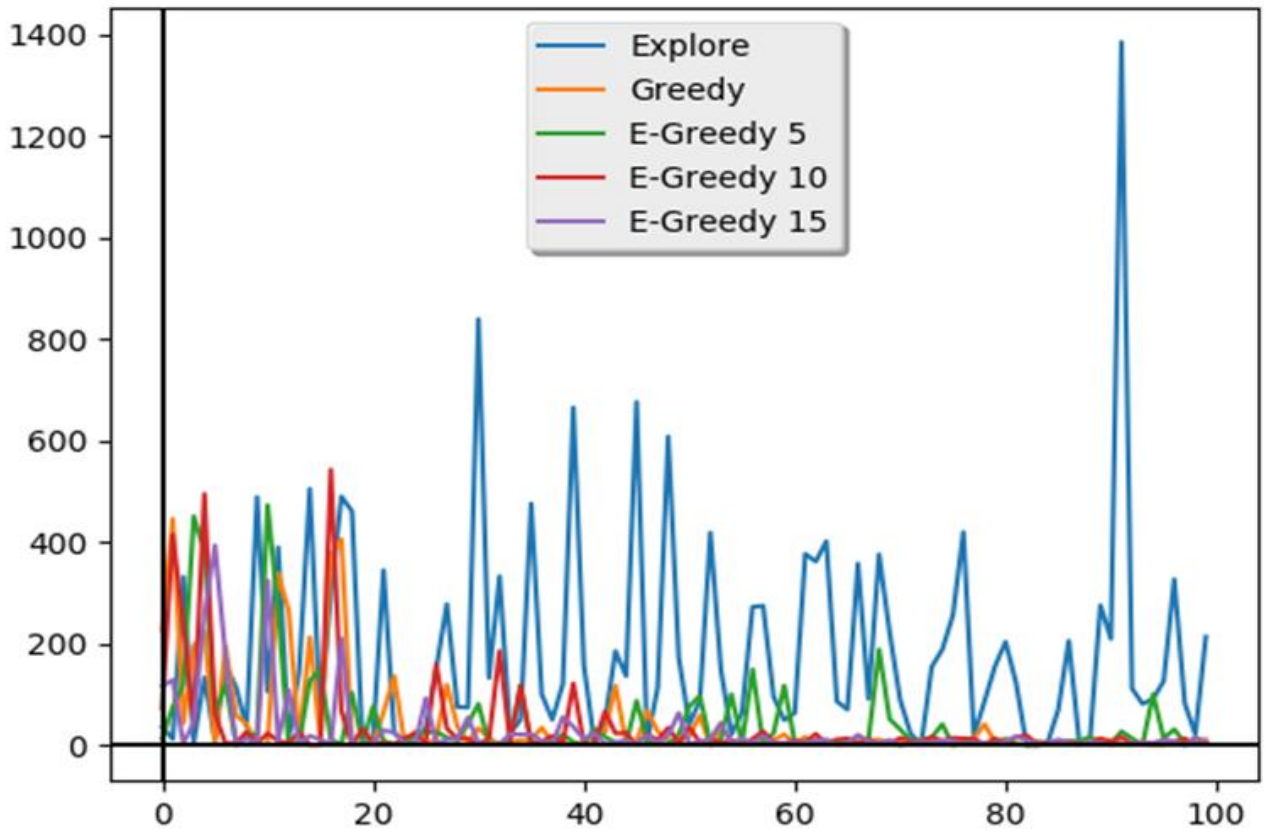
Breve explicación del trabajo realizado

En esta práctica, lo único que hemos hecho es plantear las distintas políticas que se nos plantean (medida en que se explota o se explora). Asimismo, en el caso de la política básica (solo exploración) siempre se escoge una acción aleatoria según el estado en el que te encuentres, todo hasta llegar al objetivo. Esta parte ya se nos presentaba hecha, y lo único que se hizo fue añadirle un contador para el número de acciones totales (todos los episodios) y el número de acciones en cada uno de los 100 episodios (se podrían hacer más episodios o menos, en principio es algo arbitrario).

Para las otras políticas (greedy y e-greedy) se utilizó una única idea, estructura o función ya que el greedy es un caso particular de la política e-greedy (cuando el porcentaje de exploración es cero). De esta manera, en este tipo de política, dado un porcentaje, se exploran posibilidades en esa proporción y el resto de las veces se explota. En nuestro caso los porcentajes de exploración usados fueron: 0 (greedy), 5, 10 y 15.

Resultados

A continuación se muestra una grafica donde reflejamos el número de acciones en cada uno de los episodios para la política básica (solo exploración) y para los distintos e-greedy previamente comentados (con esos cuatro porcentajes).



A continuación mostramos en una tabla los valores correspondientes a la media de acciones por episodio para cada política (en distintas ejecuciones del programa). La fila resaltada corresponde a los valores de salida tomados en la misma ejecución que la grafica anterior.

Explore	Greedy	E-Greedy 5	E-Greedy 10	E-Greedy 15
185.71	44.02	44.14	35.29	28.27
235.54	37.69	27.65	38.65	43.82
205.82	29.46	38.21	53.1	45.58
174.75	66.55	34.14	30.9	36.34
210.34	35.39	24.26	26.24	33.76
269.94	37.01	30.23	43.31	34.53

Cabe añadir que como existe bastante aleatoriedad en el proceso, los valores para cada episodio son bastante variables y existen bastantes picos. Aunque lo más recomendable para suavizar esto sería sacar medias del número de acciones a medida que se avanza de episodio, se decidió poner tal cual porque fue lo que se mostró en la defensa y se puede observar lo que buscamos comentar.

En esta gráfica podemos apreciar que en el caso del “Explore” no se refleja ningún tipo de aprendizaje ya que siempre se explora de manera aleatoria. No obstante, aunque no se refleje el aprendizaje, la tabla Q (donde almacenamos información interesante para alcanzar el objetivo; nuestro conocimiento) está completa o casi completa cuando acaban todos los episodios. Al limitarse exclusivamente a la exploración, no se observa una disminución de acciones en la gráfica (por mucho conocimiento que adquiere no lo explota en ningún momento).

Por el contrario, para las distintas versiones de la política greedy si vemos reflejado la aplicación del conocimiento. Esto lo vemos porque el número de acciones varia (disminuye) a medida que pasan los episodios. Lo cierto es que en esto no hay nada absoluto y hay que tratar de buscar un cierto equilibrio en el porcentaje dedicado a la exploración y el dedicado a la explotación para evitar la no explotación del conocimiento y para evitar la explotación del conocimiento sin “abrir nuevas posibilidades” que pueden ser mejores que la que actualmente estamos explotando.