UNIVERSIDAD POLITÉCNICA DE MADRID

ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA DE SISTEMAS INFORMÁTICOS

Máster en Software de Sistemas Distribuidos y Empotrados

Servicios y Protocolos de Aplicaciones en Internet

Clasificador de Botnet Paralelizado: Informe Adicional

Alejandro Casanova Martín

N.º de matrícula: bu0383

Madrid, 16 de octubre 2023

Ajuste del "Learning Rate"

Se entrenó el modelo utilizando diferentes valores de **alfa** (el *learning rate*, o ritmo de aprendizaje), para estudiar su impacto en el proceso de aprendizaje. En la Tabla 1 se muestra el valor del coste durante el entrenamiento, a lo largo de las iteraciones y por cada valor de alfa. En la Figura 1 se muestran las curvas de coste para cada valor de alfa (izquierda) así como el coste en la décima iteración en función de alfa (derecha).

	0.01	0.10	0.20	0.50	1.00	1.20	1.50	2.00	5.00	10.00	100.00
0	1.494698	1.351223	1.140101	1.287726	1.105428	1.204861	1.369294	1.504804	1.678974	1.534624	1.509628
1	1.488677	1.300181	1.038851	1.085837	0.643966	0.732899	0.742824	0.529281	0.294109	0.446195	3.365055
2	1.482677	1.251083	0.948047	0.920592	0.452935	0.510448	0.459344	0.341098	0.213037	0.286970	2.276020
3	1.476699	1.203891	0.867167	0.787208	0.369880	0.400590	0.345273	0.281232	0.191326	0.224109	1.665398
4	1.470743	1.158564	0.795574	0.680927	0.324747	0.340523	0.293092	0.251669	0.182592	0.196851	1.400176
5	1.464808	1.115062	0.732546	0.596927	0.296312	0.304135	0.264232	0.233928	0.177702	0.182637	1.219449
6	1.458895	1.073344	0.677306	0.530643	0.276621	0.280015	0.245889	0.222021	0.174510	0.174543	1.088198
7	1.453004	1.033370	0.629053	0.478143	0.262084	0.262872	0.233122	0.213424	0.172248	0.169772	0.988399
8	1.447134	0.995101	0.586988	0.436259	0.250854	0.250038	0.223674	0.206890	0.170559	0.166875	0.914527
9	1.441286	0.958496	0.550337	0.402534	0.241879	0.240044	0.216375	0.201733	0.169246	0.165050	0.858822

Tabla 1. Coste en función del valor de alfa (columnas) y el número de iteración (filas).

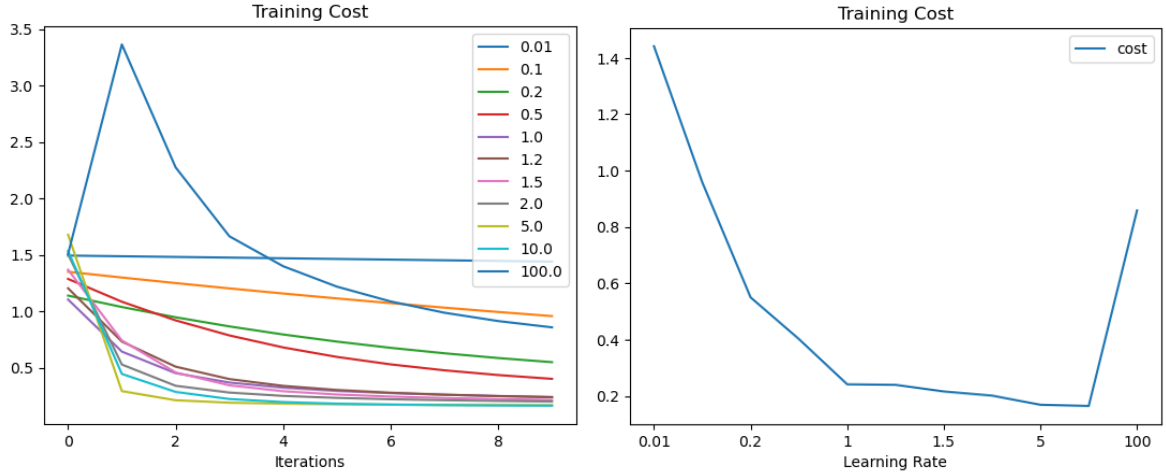

Figura 1. Izquierda: Curvas de coste, en función del número de iteración, para cada valor de alfa. Derecha: coste en la última iteración frente al valor de alfa.

Analizando los resultados podemos observar que el valor de alfa afecta notablemente al proceso de entrenamiento. Sabemos que alfa multiplica a la derivada de los pesos de nuestro modelo, a la hora de actualizar éstos mediante el algoritmo del gradiente descendente; y que por lo tanto su magnitud afectará directamente a la velocidad con la que el algoritmo converge. Esto es precisamente lo que observamos en los resultados. Para valores de alfa muy pequeños, el coste disminuye muy lentamente, casi de forma lineal; mientras que a medida que alfa aumenta, las curvas de coste adquieren una forma más asintótica, alcanzando antes valores de coste más pequeños. Sin embargo, para valores de alfa demasiado altos, el algoritmo se vuelve inestable. En nuestro caso, para alfa=100 el algoritmo sobre-oscila y luego converge, aunque para valores mayores podría incluso no estabilizarse. Además, para alfa=100, el coste en la iteración 10 es peor que en la mayoría de los casos.

Observando la tabla y las gráficas podemos concluir que, de los valores elegidos para alfa, el óptimo se encuentra entre 5 y 10, dado que el algoritmo converge antes y se logran mejores resultados.

En la Figura 2 se observan el coste y la exactitud del modelo al final del entrenamiento, en función de alfa. Se puede comprobar de nuevo que el valor óptimo de alfa se encuentra entre 5 y 10.

	cost	acc
0.01	1.441286	38.49
0.1	0.958496	53.35
0.2	0.550337	72.37
0.5	0.402534	84.08
1	0.241879	92.78
1.2	0.240044	92.07
1.5	0.216375	92.93
2	0.201733	93.20
5	0.169246	93.72
10	0.165050	93.71
100	0.858822	93.41

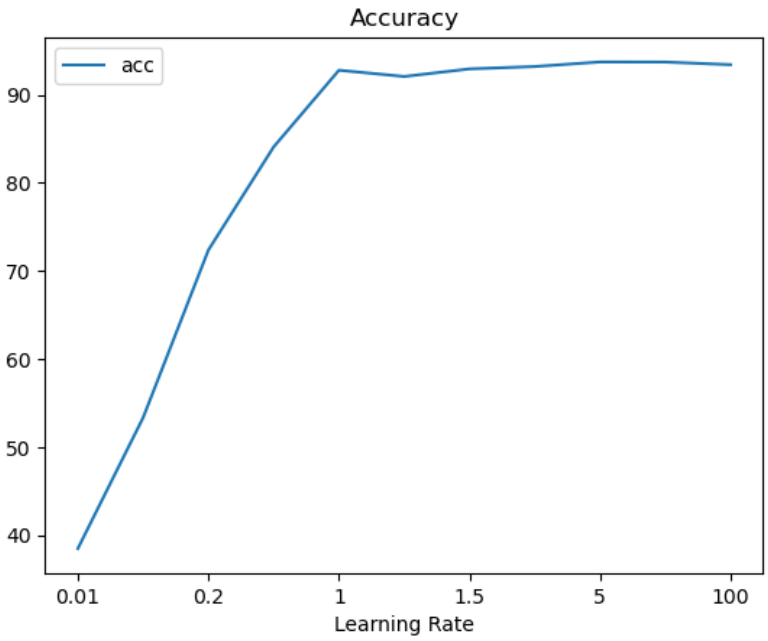

Figura 2. Izquierda: Coste y Exactitud (accuracy) en la última iteración, para cada valor de alfa. Derecha: exactitud en la última iteración en función de alfa.

Análisis del Rendimiento

Dado que hemos implementado el algoritmo de entrenamiento de forma paralelizada, cabe esperar que al aumentar el número de nodos o trabajadores (*workers*), el rendimiento del algoritmo aumente (o lo que es lo mismo: que el tiempo de entrenamiento disminuya). Dado que ejecutamos el algoritmo localmente en una única máquina, se ha realizado esta comprobación modificando el número de núcleos de procesador utilizados (en nuestro caso: de 1 a 4). En la Tabla 2 se observa el rendimiento en segundos y la aceleración (rendimiento con 1 núcleo / rendimiento) en función del número de núcleos utilizados, y en la Figura 3 se representan ambas medidas (rendimiento a la izquierda y ratio de aceleración de éste a la derecha).

Como cabría esperar, al aumentar el número de trabajadores, aumenta también el rendimiento de nuestro algoritmo. Sin embargo, dicho aumento no es exactamente lineal; al cuadriplicar el número de trabajadores, el tiempo de entrenamiento no se reduce a un cuarto, sino a poco menos de la mitad. Esto probablemente se deba a que hay ciertos tiempos (por ejemplo, los de gestión de la memoria compartida, o los de coordinación entre los trabajadores y el *driver* de *Spark*) que no pueden reducirse aumentando el número de trabajadores.

	performance	speedup
1	149.153615	1.000000
2	99.243128	1.502911
3	81.536874	1.829278
4	64.897206	2.298306

Tabla 2. Rendimiento (en segundos) y ratio de aceleración de nuestro algoritmo de entrenamiento, en función del número de núcleos.

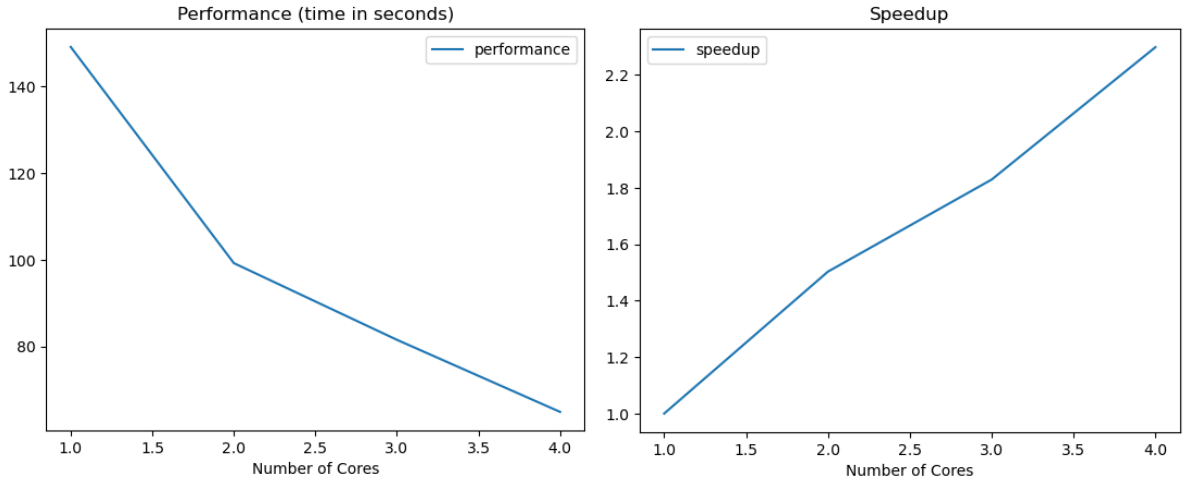

Figura 3. Izquierda: rendimiento en segundos en función del número de núcleos. Derecha: ratio de aceleración en función del número de núcleos.