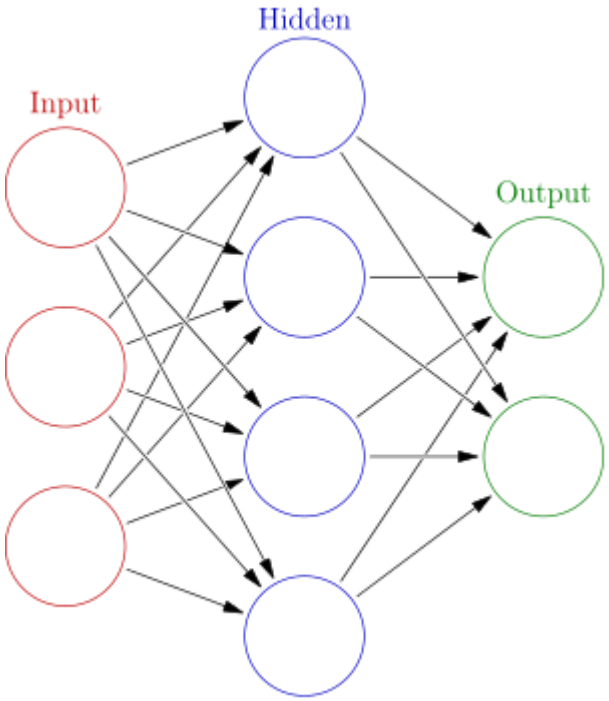
Red neuronal montada en un cluster

Alumno:
Enrico **Cagnazzo**

Descripción idea inicial

Una red neuronal necesita de mucha potencia de cálculo para poderse ejecutar sobre todo para cantidad de datos grandes. Pero es también facil poder paralelizar una red neuronal para explotar diferentes computadores porque su funcionamiento es muy estratificado.

El tipo de red neuronal elegida para el cluster es el Multi Layer Perceptron (MLP). La red elegida es esta porque tiene solo tres niveles (el mismo número de servidores del cluster) y también porque es un tipo sencillo y al mismo tiempo bastante eficaz (una red de tipo Convolutional Neural Network es mucho más compleja de programar).



Ejemplo de red neuronal de tipo MLP

Descripción de las redes neuronales

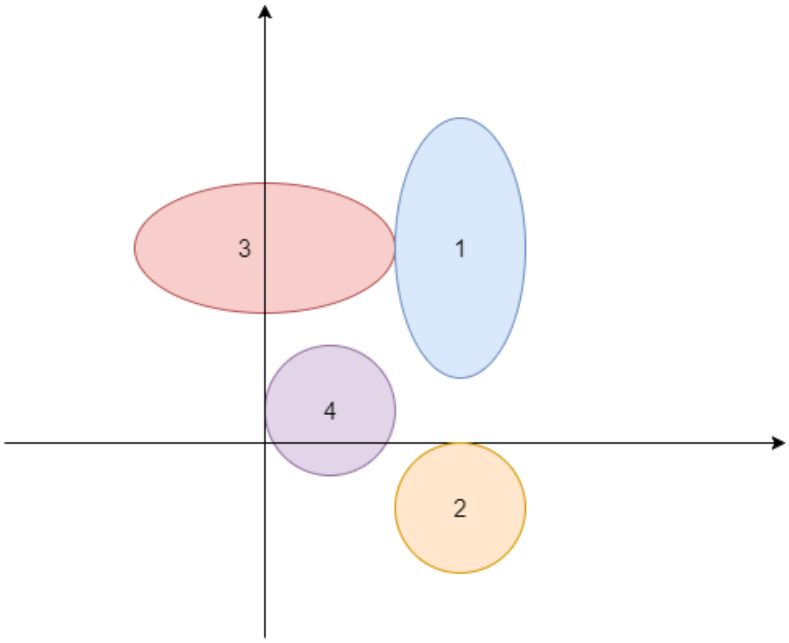
En una red neuronal hay diferentes niveles de neuronas y cada neurona está conectada con otras mediante arcos unidireccionales pesados.

El valor de las neuronas es equivalente a la suma de los valores entrantes, donde para valor entrante se entiende el producto entre la neurona conectada con el arco y el peso del mismo arco. Además se pueden aplicar funciones matemáticas al valor final de las neuronas.

Problema examinado

La red neuronal creada quiere ser un clasificador de cluster de puntos, que sea capaz de determinar a cual cluster pertenece cada punto sin haber ninguna información sobre los cluster.

Los valores iniciales serán los valores de la coordenadas ‘x’ i ‘y’ y el resultado final será un vector con la probabilidad que el punto pertenezca al cluster correspondiente.



Los cluster utilizados en el problema

Solución

Para poder distinguir de manera más fácil los puntos de los cluster se han aumentado las dimensiones del dominio tramite operaciones no lineales que permitirán a la red neuronal crear superficies que separan los puntos.

En total la red neuronal habrá 16 neuronas agrupados en la siguiente manera:

- 6 en el nivel de input: los primeros dos abran las coordenadas ‘x’ i ‘y’ y los otros cuatros el resultado de las operaciones no lineares;
- 6 en el nivel oculto: el nivel oculto es necesario para obtener obtener superficies no planas y mejorar las soluciones;
- 4 en el nivel de salida: cada neurona corresponde a un determinado cluster (prima neurona → primo cluster, segunda neurona → segundo cluster, ...) y el valor será la probabilidad que el punto pertenezca a aquel cluster.

Como funciones de activación se han utilizado la función ReLU (que permite de haber un aprendizaje más eficiente) en el nivel oculto y la función softMax en el

nivel de salida (que haz de manera tal que cada neurona tenga un valor entre 0 y 1 y que la suma de todos es 1).

Para simplificar el problema se ha quitado la faz de aprendizaje porque necesitaba de otros dos servidores para no bloquear todo el proceso de paralelización. Además trabajando con servidores virtuales en un portátil añadir otros dos podría ser muy dispendioso para el ordenador y ralentizarlo demasiado.

Implementación de la solución

Sera el master a lanzar la computación de cada punto mientre en los tres servidores se ejecutarán calculos diferentes en manera tal que cada server corresponda a un dado nivel de la red.

Las tareas de cada servidor

Tipo de paralelismo

En esta manera se ha intentado recrear un paralelismo a nivel de instrucción creando una pipeline de instrucciones: cuando un servidor acaba su computacion con un punto guarda los resultados y empieza a calcular los datos siguientes.

	Master	Servidor 1	Servidor 2	Servidor 3	
2	1				
3	2	1			
4	3	2	1		
5	4	3	2	1	
6	5	4	3	2	1

Esquema de elaboración de los datos

Almacenamiento

En el NFS ha sido creada una carpeta para toda la red neuronal, en ella están la diferentes subcarpetas para separar los ficheros de acuerdo con sus utilizo.

Codigo

En esta carpeta están todos los ficheros C++ necesarios para el funcionamiento de la red neuronal (desde la creación del dataset hacia la corrección de los pesos).

Compiled

En esta carpeta están los ficheros C++ compilados, ha sido creada para una cuestión de facilidad de uso y no equivocarse entre el código C++ y los ejecutables.

Dataset

Aqui estan todos los ficheros: los iniciales que contienen las coordenadas ‘x’ i ‘y’ y todos los creados de los diferentes servidores.

Scripts

El orden con el cual los servidores son llamados para computar los datos es gestido enteramente desde scripts, también tramite los scripts se gestiona el proceso de coleccionar los resultados y, si fuera activa, la faz de aprendizaje.

Weights

En esta carpeta son presentes los dos ficheros donde están guardados los pesos de los arcos que conectan las neuronas entre ellas. Ademas aqui serian guardados los ficheros necesario para corregir los pesos a medida que los datos son computados.

Red

Los servidores comunican entre ellos usando una red propia para evitar de obstruir la red principal interna porque es necesario un uso muy grande del comando “ssh” que puede ser muy dispendioso en términos de uso de la misma red.

Todavia para recuperar los ficheros desde el NFS los servidores siguen utilizando la red interna para no crear un otro raid compartido en la segunda red y no haber problemas de montajes de los raid en los servidores.

Resultados

El programa se ha ejecutado en las dos modalidad (secuencial y paralela) con 400 puntos en input.

También si se han utilizado 5 servidores al mismo tiempo en la modalidad paralela el speedup máximo alcanzable serìa poco superior a 3 puntos porque de estos 5 servidores uno actúa en manera síncrona con los otros (las interacciones con el servidor NFS para leer y escribir los ficheros son síncronas y por esto no se puede ganar tiempo) y uno (el master) hace una tarea mucho más pequeña de las otras tres.

La media del tiempo necesario en el sistema secuencial es de 5 minutos y 10 segundos mientras para el sistema paralelo han sido suficientes 2 minutos y 30 segundos para procesar todos los puntos de input.

El speedup resulta ser 2,1, este dato es debido a diferentes factores:

- Los servidores son virtuales y montados en un ordenador de mesa, en la simulación del cálculo paralelo la CPU ha llegado a un 100% de uso y había también sobrepasado la frecuencia de reloj normal. Esto significa che la CPU del ordenador no aguantaba a repartir todo los cálculos en paralelo;
- La complejidad de las cuatros fases no es igual entonces no se ha podido haber una partición perfecta del cómputo y la fase más pesada en términos computacionales has actuado como un cuello de botella disminuyendo la eficacia de la paralelización.

Dado que se ha quitado la parte de aprendizaje de la red neuronal para una falta de los servidores no tiene sentido enseñar los resultados en términos de capacidad de la red neuronal a clasificar los diferentes puntos en los cluster de pertenencia.

Problemas encontrados

Unas causas que han ralentizado el sistema son las siguientes

SSH

Cada pasaje desde un servidor hacia el otro necesita que una conexión ssh sea creada y esto no es instantáneo, de hecho es esta una de las causas principal que ralentiza todo el sistema quitando tiempo de computación a los cálculos de la red neuronal.

Una posible solución sería aquella de establecer una conexión con el servidor siguiente sin destruirla. Para hacer esto es necesario también un sistema que a través segnales de interrupt permiten mover la elaboracion hacia adelante.

Recuperar los ficheros

Sin embargo leer y escribir ficheros sobre la misma máquina en la cual se está ejecutando el código es mucho más rápido que hacerlo en otra porque se ahorra el tiempo de conexión entre los servidores.

Para solucionar, de manera parcial, esto se puede hacer en manera que las conexiones entre máquinas para la lectura de los ficheros sean las menores posibles.

Por ejemplo se podrían poner los ficheros de los pesos directamente en los servidores 2 y 3 porque estos son los únicos que los necesitan.

Además cada servidor podria crear los ficheros de los resultados directamente en el servidor que los utilizaran (ej: el servidor 1 crea directamente en el servidor 2) de manera tal de reducir a la mitad el número de conexiones necesarias para la lectura y escritura de los resultados parciales.

Una otra solucion es de poner el resultado de cada computacion directamente como parámetro de las llamadas al servidor seguente, en esta manera además que quitar la necesidad de conectarse a otra máquina para leer y/o escribir los resultados. De esta manera se obtenía un vantage en términos de prestaciones (minor) también con un sistema secuencial

Conclusiones

Sin embargo hay maneras mejores para paralelizar la computación de una red neuronal y me habría gustado mucho intentar de hacer la misma cosa con una tarjeta grafica (sobretodo utilizando el sistema de rCuda) pero el ordenador que uso no tiene una tarjeta Cuda y la que hay (de AMD) no era posible conectarla con los servidores virtuales ni utilizarla trámite sistemas remotos como rCuda.

A la luz de estos resultados se puede concluir que el sistema de paralelización entre si trabaja bien pero para explotar toda la potencia de computación en la manera mejor sería necesario cambiar unas cosas en la implementación o también utilizarlo en calculo muchos más largos así que el tiempo necesario para conectarse entre las máquinas (para utilizar ssh y gestionar ficheros de otras maquinas) diventa irrelevante respecto el tiempo de cálculo.