DEPARTAMENTO
DE COMPUTACION
Facultad de Ciencias Exactas y Naturales - UBA

Trabajo Final

30 de julio de 2017 — Organización del Computador II

Integrante	LU	Correo electrónico
Costa, Manuel	COMPLETAR	COMPLETAR
Gatti, Mathias	477/14	mathigatti@gmail.com

Facultad de Ciencias Exactas y Naturales
Universidad de Buenos Aires
Ciudad Universitaria - (Pabellón I/Planta Baja)
Intendente Güiraldes 2160 - C1428EGA
Ciudad Autónoma de Buenos Aires - Rep. Argentina
Tel/Fax: (54 11) 4576-3359
http://www.fcen.uba.ar

Índice

0. Introducción

La idea de este trabajo fue implementar una red neuronal sigmoidea. Debido a la intensa cantidad de computos que podían realizarse de forma paralela (Principalmente operaciones matriciales) pensamos como foco principal del proyecto estudiar que tanto se podría mejorar la performance temporal del programa aprovechando SIMD y demás beneficios que podemos sacar a partir de la utilización de ASSEMBLER.

Decidimos que la red resuelva la clásica tarea de reconocimiento de digitos manuscritos, utilizando el dataset MNIST. Escogimos esta tarea por ser un problema de tamaño razonable para tratar con el hardware del que disponemos, pero que a su vez es lo suficientemente grande como para permitir un análisis interesante.

1. Manual de usuario

1.1. Uso del programa

El programa esta formado por un set de datos sobre el cual trabajamos llamado MNIST el cual contiene unas matrices de números enteros las cuales representan imágenes de números centrados y de tamaño fijo. Estas son de 28 pixeles de ancho por 28 pixels de alto en escala de grises.

Este set de datos esta en la carpeta *mnist* y es convertido a archivos de texto que contienen un pixel por línea con un script de python el cual aloja los archivos en la carpeta *data*. Esto se hace para facilitar la posterior lectura de estos datos por nuestro programa.

Una vez hecho esto se pueden compilar y ejecutar nuestros programas que crean y entrenan una red neuronal para reconocer estos caracteres. Tanto en la carpeta $float$ como en la carpeta *double* se encontrará el código que implementa dicho programa y las correspondientes herramientas de compilación. Para ejecutar el programa basta compilarlo con el `Makefile` y luego ejecutar `asm_version` o `c_version` según la versión que uno desee probar.

1.2. Tests

Tanto la versión del programa que recibe Floats como la que recibe Doubles tiene una batería de tests en las funciones que implementamos tanto en ASSEMBLER como C, esto fue para corroborar el buen funcionamiento de lo dessarrollado y ayudarnos a debuguear.

Los test se compilan con el `Makefile` el cual crea el ejecutable test que corre los mismos.

2. Tópicos Teóricos

–COMPLETAR CON TOPICOS TEORICOS–

3. Resultados

3.1. Porcentaje de aciertos de las redes

La red neuronal tiene un porcentaje de acierto del 95 % en todas sus versiones. A pesar de que el tipo float tiene una menor capacidad de representación numérica que su contraparte el tipo double su precisión parece ser suficiente para realizar los cálculos necesarios y hacer que la red neuronal llegue a las mismas conclusiones que la red que utiliza double.

3.2. Tiempos

Aquí se puede observar un cuadro de resultados comparando nuestras implementaciones de ASSEMBLER contra las de C compiladas con gcc -03.

Versión	cost_derivative	mat_plus_vec	update_weight	hadamard_product	matrix_prod
Double C	0.019694	0.323289	0.298969	0.966852	0.841608
Double ASM	0.019881	0.338240	0.758107	0.341586	0.498035
Float C	0.019256	0.171635	0.158144	0.922496	0.711830
Float ASM	0.018904	0.170835	0.381136	0.167613	0.490826

Tabla 1 Cuadro de resultados obtenidos en cada método.

4. Conclusiones

–COMPLETAR CONCLUSION–