Técnicas de Aceleración por Hardware para Algoritmos de Aprendizaje de Máquina

Alejandro Piad-Morffis
Suilan Estévez-Velarde
Yudivián Almeida-Cruz

Grupo de Inteligencia Artificial
Facultad de Matemática y Computación
Universidad de La Habana

EMNO 2015

Procesos más costosos

Procesos más costosos

Aprendizaje de máquina

- Entrenamiento
- Clasificación
- Ajuste de parámetros

Procesos más costosos

Aprendizaje de máquina

- Entrenamiento
- Clasificación
- Ajuste de parámetros

Optimización (algoritmos iterativos)

- Evaluación de la función objetivo
- Operaciones algebraicas

Técnicas de aceleración

Alternativas

- Algoritmos más eficientes
- Lenguajes de bajo nivel
- Paralelización a nivel de hilos
- Distribución de cómputo
- Programación en GPU

Técnicas de aceleración

Alternativas

- Algoritmos más eficientes
- Lenguajes de bajo nivel
- Paralelización a nivel de hilos
- Distribución de cómputo
- Programación en GPU

Programación en GPU

- Paralelismo de datos
- Sincronización
- Transferencia CPU-GPU
- Modelo de cómputo no Turing-completo

Software

Plataformas

- OpenCL
- CUDA
- DirectCompute

Bibliotecas

- **C, C++**: CUDA Toolkit, OpenCL, OpenCV
- **R**: RPud, RPuSvm, RPudPlus
- **MATLAB**: Parallel Computing Toolbox
- **Python**: PyCUDA, PyOpenCL, Theano, GNumpy
- **.NET**: DirectCompute, ExtremeOptimization

Experimentación

Operaciones

- Funciones trigonométricas
- Funciones exponenciales
- Álgebra lineal

Experimentación

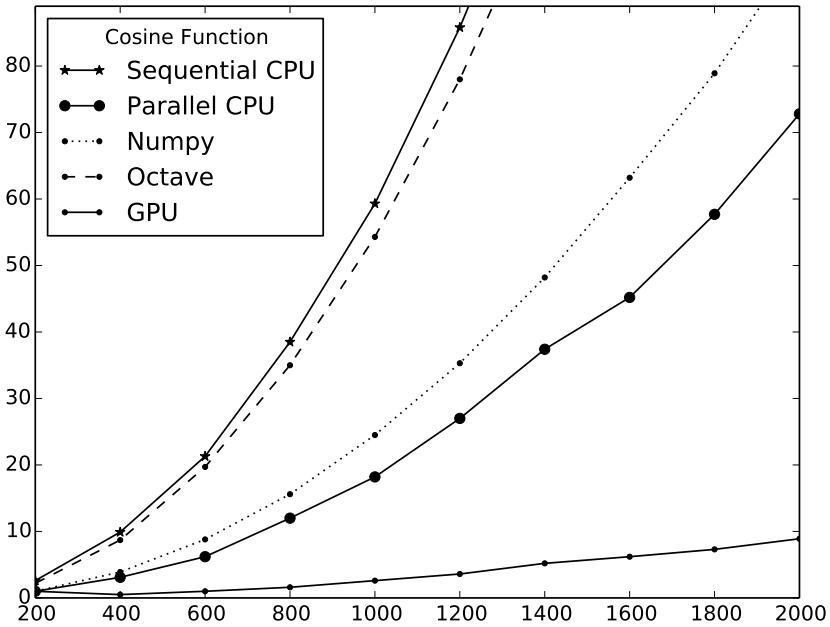
Operaciones

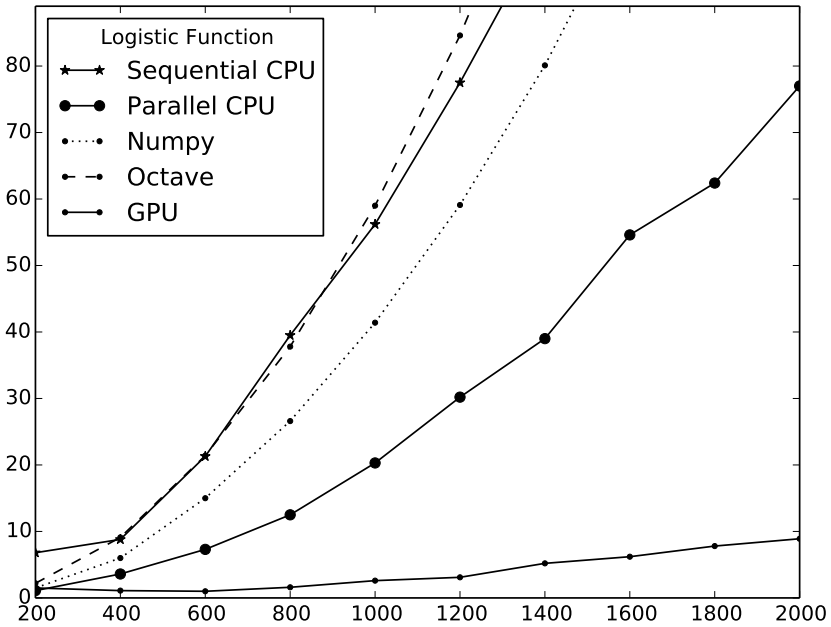
- Funciones trigonométricas
- Funciones exponenciales
- Álgebra lineal

Implementaciones

- Secuencial (.NET)
- Paralelismo de hilos (.NET)
- Octave
- Numpy
- Paralelismo en GPU (DirectCompute)

Funciones elementales

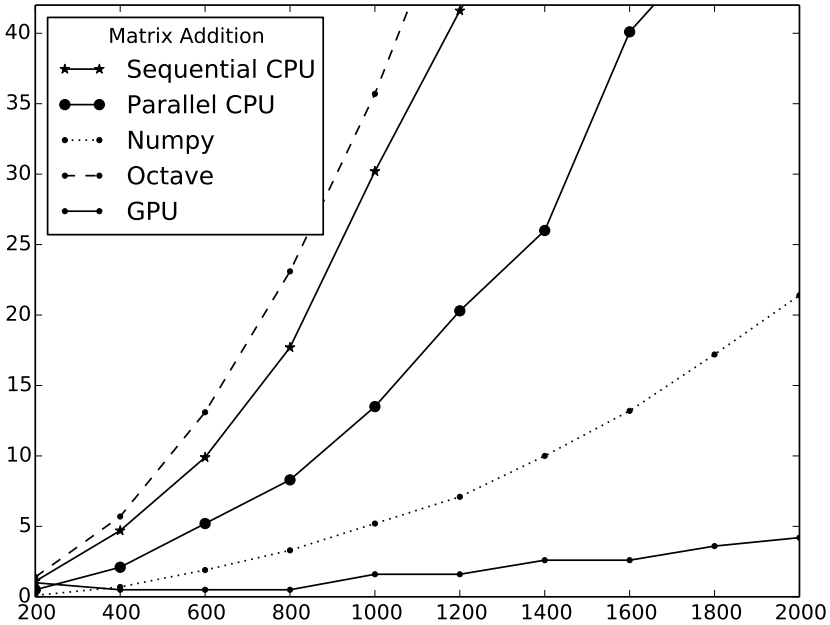


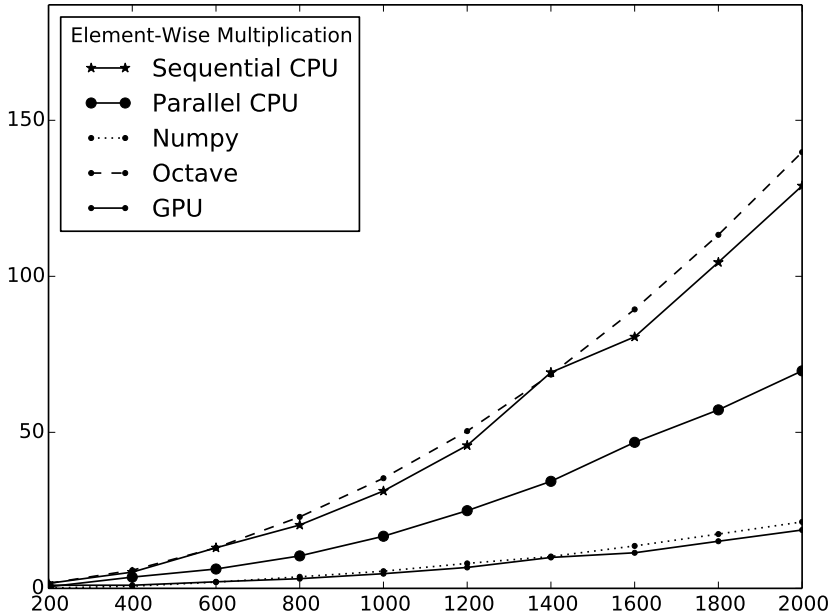


$$[y_{i,j}]_{n,m} = [\cos(x_{i,j})]_{n,m}$$

$$[y_{i,j}]_{n,m} = \left[\frac{1}{1+e^{-x_{i,j}}}\right]_{n,m}$$

Algebra lineal (elemento a elemento)

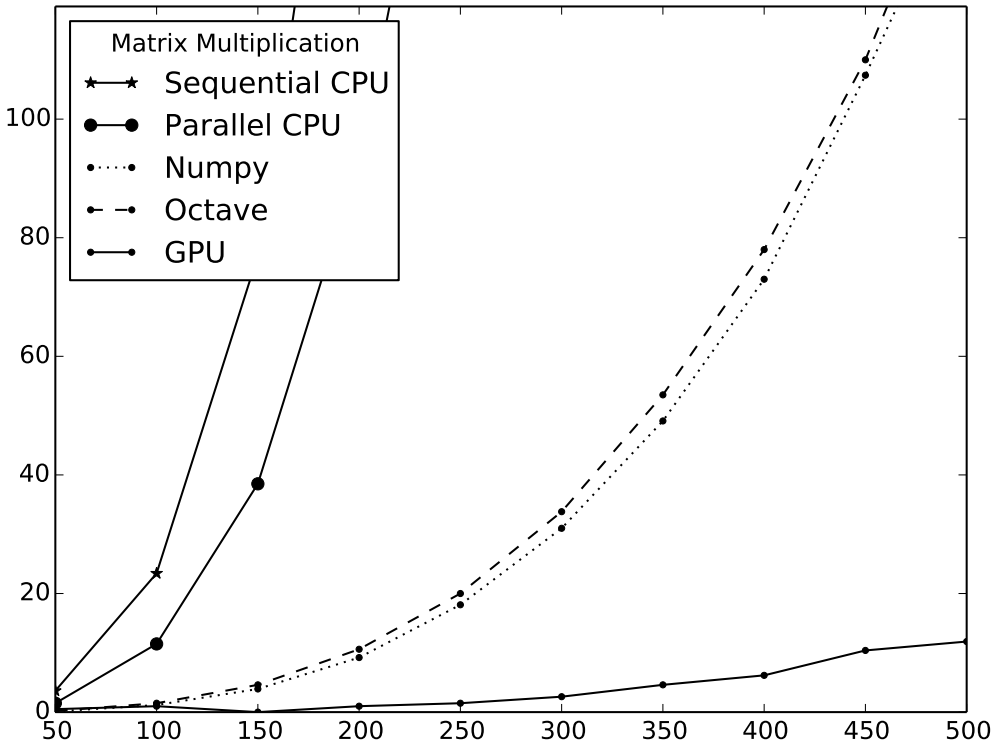




$$[z_{i,j}]_{n,m} = [x_{i,j} + y_{i,j}]_{n,m}$$

$$[z_{i,j}]_{n,m} = [x_{i,j} \times y_{i,j}]_{n,m}$$

Algebra lineal (multiplicación de matrices)



$$[z_{i,j}]_{n,m} = \left[\sum_k x_{i,k} \times y_{k,j} \right]_{n,m}$$

Backpropagation

- Dimensión: 500; 5 capas; 1000 iteraciones
- Lotes de tamaño 500
- Función logística (500×500)
- Multiplicación y adición de matrices (500×500)

Algoritmo	**Tiempo**	**%-diff**
Sequential CPU	7h 48m 28s	99.6
Parallel CPU	3h 31m 48s	99.2
Octave	0h 22m 13s	91.9
Numpy	0h 20m 08s	91.1
GPU	0h 01m 48s	-

Función Rastrigin en CEC-2013

$$f(\vec{x}) = An + \sum_{i=1}^{n} \left[x_i^2 - A\cos\left(2\pi x_i\right) \right]$$

- Población: 1000; dimensión: 1000; 10,000 iteraciones
- Multiplicación elemento a elemento (1000×1000)
- Función coseno (1000×1000)
- Adición de matrices (1000×1000)

Algoritmo	**Tiempo**	**%-diff**
Octave	0h 20m 53s	92.9
Sequential CPU	0h 20m 07s	92.6
Parallel CPU	0h 08m 04s	81.6
Numpy	0h 05m 52s	74.7
GPU	0h 01m 29s	-

Conclusiones

Usos recomendados

- Algoritmos iterativos (misma función de evaluación)
- Grandes dimensiones
- Grandes volúmenes de datos
- Código vectorial

Conclusiones

Usos recomendados

- Algoritmos iterativos (misma función de evaluación)
- Grandes dimensiones
- Grandes volúmenes de datos
- Código vectorial

Paralelismo en GPU

- Recursos de cómputo baratos y disponibles
- Complejidad de implementación (bibliotecas)
- Considerable incremento del rendimiento

Técnicas de Aceleración por Hardware para Algoritmos de Aprendizaje de Máquina

Alejandro Piad-Morffis
Suilan Estévez-Velarde
Yudivián Almeida-Cruz

Grupo de Inteligencia Artificial
Facultad de Matemática y Computación
Universidad de La Habana

EMNO 2015