GPU Parallel AIME

Ivan Au Yeung, Senior Solutions Architect | 2nd Sep 2025

Library comparison for pinv

Numpy vs Pytorch vs Cupy

- **linalg.pinv Library Comparison:**

CuPy and PyTorch offer significant speedups (4.5x and 4.24x respectively) for pinv with only a modest reduction in accuracy.

- **Batched Computation:**

PyTorch enables parallel batched computation of the inverse, delivering a substantial speed advantage(6.73x) over CuPy, suitable for scenarios requiring multiple independent inversions.

```
================================================================================
COMPREHENSIVE PERFORMANCE AND ACCURACY COMPARISON
================================================================================
Method                    Time (s)   Speedup vs M1 Error         Status
--------------------------------------------------------------------------------
Method 1 (NumPy pinv)     1.3569     1.00          8.87e-16     PASSED
Method 2 (NumPy solve)    0.1393     9.74          2.24e-15     PASSED
Method 3 (PyTorch pinv)   0.3203     4.24          6.95e-05     PASSED
Method 4 (PyTorch solve)  0.0340     39.90         6.81e-05     PASSED
Method 5 (PT batch pinv)  0.2018     6.73          6.95e-05     PASSED
Method 6 (PT batch solve) 0.0016     866.97        6.82e-05     PASSED
Method 7 (CuPy pinv)      0.3015     4.50          8.24e-07     PASSED
Method 8 (CuPy solve)     0.0073     185.20        4.83e-07     PASSED
--------------------------------------------------------------------------------
```

Library comparison for pinv

Numpy vs Pytorch vs Cupy

- **linalg.solve formulation:**

New formulation explaining why the `linalg.solve` method is appropriate for solving the system in equation

Original formulation for $A^\dagger$:

$$
\begin{aligned}
X &= A^\dagger \widehat{Y}, \\
X\widehat{Y}^T &= A^\dagger \widehat{Y}\widehat{Y}^T, \\
X\widehat{Y}^T \left(\widehat{Y}\widehat{Y}^T\right)^{-1} &= A^\dagger \left(\widehat{Y}\widehat{Y}^T\right)\left(\widehat{Y}\widehat{Y}^T\right)^{-1}, \\
A^\dagger &= X\widehat{Y}^T \left(\widehat{Y}\widehat{Y}^T\right)^{-1} = X\widehat{Y}^\dagger
\end{aligned}
$$

Formulation for using *linalg.solve*:

$$
\begin{aligned}
X &= A^\dagger \widehat{Y}, \\
X\widehat{Y}^T &= A^\dagger \widehat{Y}\widehat{Y}^T, \\
X\widehat{Y}^T \left(\widehat{Y}\widehat{Y}^T\right)^{-1} &= A^\dagger \left(\widehat{Y}\widehat{Y}^T\right)\left(\widehat{Y}\widehat{Y}^T\right)^{-1}, \\
A^\dagger &= X\widehat{Y}^T \left(\widehat{Y}\widehat{Y}^T\right)^{-1} \\
(A^\dagger)^T &= \left(\left(\widehat{Y}\widehat{Y}^T\right)^{-1}\right)^T \left(X\widehat{Y}^T\right)^T \\
(A^\dagger)^T &= \left(\widehat{Y}\widehat{Y}^T\right)^{-1} \widehat{Y}X^T
\end{aligned}
$$

COMPREHENSIVE PERFORMANCE AND ACCURACY COMPARISON

Method	Time (s)	Speedup vs M1	Error	Status
Method 1 (NumPy pinv)	1.3569	1.00	8.87e-16	PASSED
Method 2 (NumPy solve)	0.1393	9.74	2.24e-15	PASSED
Method 3 (PyTorch pinv)	0.3203	4.24	6.95e-05	PASSED
Method 4 (PyTorch solve)	0.0340	39.90	6.81e-05	PASSED
Method 5 (PT batch pinv)	0.2018	6.73	6.95e-05	PASSED
Method 6 (PT batch solve)	0.0016	866.97	6.82e-05	PASSED
Method 7 (CuPy pinv)	0.3015	4.50	8.24e-07	PASSED
Method 8 (CuPy solve)	0.0073	185.20	4.83e-07	PASSED

Formulation with zero-order Tikhonov regularization of regularization parameter ε:

$$
(A^\dagger)^T = \left(\widehat{Y}\widehat{Y}^T + \varepsilon I\right)^{-1} \widehat{Y}X^T
$$

Library comparison for pinv

Numpy vs Pytorch vs Cupy

- **linalg.solve Library Comparison :**

Numpy.linalg.solve offer 9.74 speedup with small accuracy reduction

CuPy and PyTorch offer significant speedups (185.2x and 39.9x respectively) for solve with only a modest reduction in accuracy.

Batch Pytorch offer even larger speedup(867x) and similar reduction in accuracy to Pytorch

COMPREHENSIVE PERFORMANCE AND ACCURACY COMPARISON

Method	Time (s)	Speedup vs M1	Error	Status
Method 1 (NumPy pinv)	1.3569	1.00	8.87e-16	PASSED
Method 2 (NumPy solve)	0.1393	9.74	2.24e-15	PASSED
Method 3 (PyTorch pinv)	0.3203	4.24	6.95e-05	PASSED
Method 4 (PyTorch solve)	0.0340	39.90	6.81e-05	PASSED
Method 5 (PT batch pinv)	0.2018	6.73	6.95e-05	PASSED
Method 6 (PT batch solve)	0.0016	866.97	6.82e-05	PASSED
Method 7 (CuPy pinv)	0.3015	4.50	8.24e-07	PASSED
Method 8 (CuPy solve)	0.0073	185.20	4.83e-07	PASSED

Library comparison for pinv

Cupynumeric vs Cupy

- **Cupynumeric Library Comparison :**

Regarding parallel computation of the pseudoinverse, the cupynumeric library currently does not support this functionality. However, `linalg.solve` is supported and can leverage multiple GPUs for processing a single matrix. You can find an example in the attached file "cpn_solve_example.py." To test multi-GPU performance, please run:

```
legate --gpus 2 cpn_solve_example.py --size 50000 --skip-cupy
```

This configuration enables parallel computation for matrices as large as 50,000 x 50,000; in contrast, CuPy encounters out-of-memory errors at this scale.

50000x50000 matrix	1 gpu (RTX 5880 ada)	2 gpu (RTX 5880 ada)	Cupy
Time (ms)	16479.9410 ms	12114.6650 ms	OOM

The End