Ko‘p o‘lchovli massiv uchun belgilangan funksiyalar va ularning zamonaviy hisoblash usullaridagi xususiyatlari

Gulbodom Oybek qizi Norqulova
BXU

Annotatsiya: Ushbu maqolada ko‘p o‘lchovli massivlar bilan ishlashda qo‘llaniladigan asosiy funksiyalar va ularning zamonaviy hisoblash tizimlari, xususan, parallel hisoblash va GPU texnologiyalarida qo‘llanilishi ko‘rib chiqiladi. Matrisali operatsiyalar, tensorli hisoblashlar va ko‘p o‘lchovli ma’lumotlarni qayta ishlash algoritmlarining samaradorligini oshirish yo‘llari tahlil qilingan. Massiv transformatsiyalari, agregatsiya funksiyalari hamda zamonaviy dasturlash tillarida mavjud bo‘lgan kutubxonalar orqali amalga oshiriladigan optimallashtirishlar batafsil ko‘rsatilgan.

Kalit so‘zlar: ko‘p o‘lchovli massiv, tensorlar, NumPy, matritsa operatsiyalari, vektorlash, parallel hisoblash, GPU, optimallash, massiv indekslash, broadkasting, ma’lumotlar qayta ishlash, hisoblash samaradorligi, multidimensional arrays

Functions defined for multidimensional arrays and their properties in modern computing methods

Gulbodom Oybek qizi Norqulova
BSU

Abstract: This article considers the main functions used in working with multidimensional arrays and their application in modern computing systems, in particular, parallel computing and GPU technologies. Ways to increase the efficiency of matrix operations, tensor calculations and multidimensional data processing algorithms are analyzed. Array transformations, aggregation functions and optimizations implemented using libraries available in modern programming languages are shown in detail.

Keywords: multidimensional array, tensors, NumPy, matrix operations, vectorization, parallel computing, GPU, optimization, array indexing, broadcasting, data processing, computing efficiency, multidimensional arrays

Kirish. Ko‘p o‘lchovli massivlar zamonaviy ilmiy hisoblashlar, sun’iy intellekt va katta hajmli ma’lumotlarni qayta ishlashda asosiy ma’lumot strukturasi hisoblanadi.

Ikki o‘lchovli (matritsa), uch o‘lchovli va undan yuqori o‘lchovli massivlar turli xil amaliy muammolarni hal qilishda keng qo‘llaniladi.

Ko‘p o‘lchovli massivlar uchun asosiy funksiyalar

1. Yaratish va initsializatsiya funksiyalari

Massivlarni yaratish uchun turli usullar mavjud: nollar bilan to‘ldirish (zeros), birliklar bilan to‘ldirish (ones), tasodifiy qiymatlar (random), identifikatsiya matritsasi (eye), belgilangan diapazon (arange, linspace). Zamonaviy kutubxonalarda bu operatsiyalar xotira samaradorligini hisobga olgan holda optimallashtirilgan.

2. Indekslash va kesish (slicing)

Ko‘p o‘lchovli massivlarda ma’lumotlarga murojaat qilish murakkab indekslash mexanizmlarini talab qiladi. Boolean indekslash, fancy indekslash va ellipsis operatori kabi usullar ma’lumotlarni tanlashda katta qulaylik yaratadi. Indekslash operatsiyalari asl massivning ko‘rinishini (view) qaytarishi mumkin, bu xotirani tejash imkonini beradi.

3. Shakl o‘zgartirish funksiyalari

Reshape, transpose, flatten, squeeze va expand_dims kabi funksiyalar massiv o‘lchamlarini o‘zgartirish uchun ishlatiladi. Bu operatsiyalar ko‘pincha nusxa yaratmasdan amalga oshiriladi va juda tez ishlaydi, chunki faqat massivning metadata ma’lumotlari o‘zgartiriladi.

4. Matematik operatsiyalar

Elementar matematik amallar (qo‘shish, ayirish, ko‘paytirish, bo‘lish) vektorlashtirilgan shaklda bajariladi. Matritsa ko‘paytmasi (dot, matmul), transpozitsiya, determinant hisoblash va teskari matritsani topish kabi chiziqli algebra operatsiyalari maxsus optimallashtirilgan algoritmlar yordamida amalga oshiriladi.

5. Agregatsiya funksiyalari

Sum, mean, max, min, std, var kabi funksiyalar massivning ma’lum o‘qi bo‘yicha yoki butun massiv bo‘yicha hisoblashlarni amalga oshiradi. Bu funksiyalar parallel hisoblash arxitekturalarida samarali ishlash uchun maxsus optimallashtirilgan.

6. Broadkasting mexanizmi

Broadkasting turli shakldagi massivlar ustida arifmetik amallar bajarish imkonini beradi. Kichik massiv avtomatik ravishda katta massiv o‘lchamiga "cho‘ziladi". Bu mexanizm xotira sarfi va hisoblash vaqtini sezilarli darajada kamaytiradi.

Zamonaviy hisoblash usullaridagi xususiyatlar

Vektorlash va SIMD

Zamonaviy protsessorlar SIMD (Single Instruction Multiple Data) texnologiyasini qo‘llab-quvvatlaydi. NumPy, PyTorch va TensorFlow kabi kutubxonalar avtomatik ravishda vektorlashtirilgan operatsiyalardan foydalanadi, bu bitta ko‘rsatma bilan bir vaqtning o‘zida ko‘plab ma’lumotlarni qayta ishlash imkonini beradi.

GPU tezlashtiruvi

CUDA va OpenCL platformalari yordamida massiv operatsiyalari grafik protsessorlar (GPU)da bajariladi. CuPy, PyTorch va TensorFlow kutubxonalari GPU-ga asoslangan hisoblashlarni qo‘llab-quvvatlaydi va minglab yadrolar parallel ishlashi tufayli hisoblash tezligi o‘nlab marta ortadi.

Xotira boshqaruvi

Zamonaviy kutubxonalar xotiradan optimal foydalanish uchun turli strategiyalarni qo‘llaydi: lazy evaluation (kechiktirilgan hisoblash), memory pooling, in-place operatsiyalar va efficient data layout (C-style va Fortran-style). Bu yondashuvlar katta hajmli ma’lumotlar bilan ishlashda muhim ahamiyatga ega.

Parallel va distributed hisoblash

Dask, Ray va Apache Spark kabi texnologiyalar ko‘p o‘lchovli massivlarni bir nechta kompyuterlar yoki protsessor yadrolari o‘rtasida taqsimlash imkonini beradi. Bu yondashuvlar xotiraga sig‘maydigan ma’lumotlar bilan ishlashda zarur.

Just-In-Time (JIT) kompilyatsiya

Numba va JAX kabi kutubxonalar Python kodini mashina kodiga kompilyatsiya qiladi, bu esa interpretatsiya xarajatlarini yo‘qotadi va hisoblash tezligini C va Fortran darajasiga yetkazadi.

Xulosa. Ko‘p o‘lchovli massivlar zamonaviy hisoblash texnologiyalarining asosiy elementi bo‘lib, ularning samarali ishlatilishi algoritm tanloviga ham, texnik platformaga ham bog‘liq. Vektorlash, GPU tezlashtiruvi, parallel hisoblash va xotira optimallashuvi birgalikda yuqori unumdorlikka erishish imkonini beradi. Kelgusida kvant hisoblash va neuromorphic arxitekturalar yangi imkoniyatlar ochishi kutilmoqda.

Foydalanilgan adabiyotlar

1. Harris C.R. et al. Array programming with NumPy // Nature. – 2020. – Vol. 585. – P. 357-362.

2. Paszke A. et al. PyTorch: An imperative style, high-performance deep learning library // Advances in Neural Information Processing Systems. – 2019. – Vol. 32.

3. Abadi M. et al. TensorFlow: Large-scale machine learning on heterogeneous systems // Software available from tensorflow.org. – 2015.

4. Lam S.K., Pitrou A., Seibert S. Numba: A LLVM-based Python JIT compiler // Proceedings of the Second Workshop on the LLVM Compiler Infrastructure in HPC. – 2015. – P. 1-6.

5. Rocklin M. Dask: Parallel computation with blocked algorithms and task scheduling // Proceedings of the 14th Python in Science Conference. – 2015. – P. 126-132.

6. Okuta R. et al. CuPy: A NumPy-compatible library for NVIDIA GPU calculations // Proceedings of Workshop on Machine Learning Systems (LearningSys)

in The Thirty-first Annual Conference on Neural Information Processing Systems. – 2017.

7. Bradbury J. et al. JAX: composable transformations of Python+ NumPy programs // Version 0.2. – 2018. – Vol. 18.

8. Van Der Walt S., Colbert S.C., Varoquaux G. The NumPy array: a structure for efficient numerical computation // Computing in Science & Engineering. – 2011. – Vol. 13. – №. 2. – P. 22-30.

9. Zaharia M. et al. Apache spark: a unified engine for big data processing // Communications of the ACM. – 2016. – Vol. 59. – №. 11. – P. 56-65.

10. Nickolls J., Dally W.J. The GPU computing era // IEEE micro. – 2010. – Vol. 30. – №. 2. – P. 56-69.