The manual for class BaseTensor

赵继泽

February 22, 2021

1 介绍

类 **BaseTensor** 中成员变量和成员函数的介绍。**BaseTensor** 定义张量以及一些基本操作。

2 类成员变量

1. **long fileSize**：张量存入磁盘的文件大小
2. **DATATYPE* tensorData**：存放数据
3. **BLASINT maxDataSize**：tensorData可以使用的最大内存

3 类成员函数

3.1 构造函数

3.1.1 BaseTensor(const char* filename, const bool &ifremoved, const int &readtype = 0) ==> 从文件中读取[reviewed]

3.1.2 BaseTensor(std::ifstream &ifs) ==> 从流中读取[reviewed]

3.1.3 template<typename T> BaseTensor(const uint32_t &tr, const BLASINT* rd, const T* data = 0, const BLASINT &dtsize = 0) ==> 从数据中构造张量[reviewed]

3.1.4 template<typename T> BaseTensor(initializer_list<BLASINT> rd, const T* data = 0, const BLASINT &dtsize = 0) ==> 构造张量，张量的维数由 initializer_list<BLASINT> 给出

3.1.5 BaseTensor(const BaseTensor &bt, const BLASINT &idx, void* tmpdata, const BLASINT &tdsize) ==> 张量和自身收缩，= T_...i..... T^*_...i.....[reviewed]

1. **const BaseTensor &bt**：即上面的张量T;
2. **const BLASINT* idx**：需要收缩的指标;
3. **void* tmpdata**：辅助空间
4. **const BLASINT &tdsize**：tmpdata 的大小(in unit of byte)，用来检查辅助空间是否够用.

3.1.6 BaseTensor(const char &cttype, const BaseTensor &btl, const BLASINT &il, const BaseTensor &btr, const BLASINT &ir, void* tmpdata, const BLASINT &tdsize) ==> 将btl中il指标和btr中ir指标收缩，收缩完以后，btl中剩下的所有指标放在左边，btr中剩下的所有指标放在右边，$\sum_{i_l}$ btl$_{...i_{l...}}$ btr$_{...j_{r(=i_l)}...}$ = $M_{...}$. btl, btr的数据最终不会发生变化，但是要注意，btl的数据在运算过程中会变化，计算完成后恢复。 注意这里没有复共轭[reviewed], 参数cttype = 'S', 'D' or 'I'.

1. 如果 **cttype = 'S'**, btl中剩下的指标放在 M 的指标的左边，btr中剩下的指标放在右边, A_ijk B_mjn = R_ikmn;
2. 如果 **cttype = 'D'**, A_ijqkl B_mnqst = R_ijmnklst;
3. 如果 **cttype = 'I'**, A_ijk B_mjn = R_imnk.

3.1.7 BaseTensor(const BaseTensor &bt, const BLASINT &il, const BLASINT &ir) ==> 内部指标收缩，$= \sum_{il}$T_...il...ir(=il)...[reviewed]

3.1.8 BaseTensor(const BaseTensor &btl, const BaseTensor &btr, void* tmpdata, const BLASINT &tdsize) ==> 张量的直积. T_(i1,j1)(i2,j2)(i3,j3)... = btl_i1i2i3.. btr_j1j2j3...

3.1.9 BaseTensor(const BaseTensor &bt, const DATATYPE &scale = 1) ==> 复制构造函数[reviewed]

1. bt：被复制的类；
2. scale：将原来的数据乘以scale

3.2 一般成员函数

3.2.1 void legProduct(const BLASINT &il, DATATYPE* lambda, const BLASINT &lmbdsize) ==> = T_....i.... * lambda[i], 不对i求和

3.2.2 void legDivision(const BLASINT &il, DATATYPE* lambda, const BLASINT &lmbdsize) ==> = T_...i.../lambda[i], 不对i求和

3.2.3 void shiftBefore(const BLASINT &il, const BLASINT &iprev, DATATYPE* tmpdata, const BLASINT &tdsize) ==> 将指标il移到指标iprev前面去；这里要求 iprev <= il [reviewed]

3.2.4 void shiftAfter(const BLASINT &il, const BLASINT &iback, DATATYPE* tmpdata, const BLASINT &tdsize) ==> 将指标il移到指标iback后面去； 这里要求 iback >= il [reviewed]

3.2.5 bool permute(initializer_list<BLASINT> ri, DATATYPE* tmpdata, const BLASINT &tdsize) ==> 和matlab中同名函数类似；

3.2.6 void tensorContraction(const char &cttype, const BaseTensor &btl, const BLASINT &il, const BaseTensor &btr, const BLASINT &ir, DATATYPE* tmpdata, const BLASINT &tdsize) ==> 张量指标的收缩，btl剩下的指标放左边，btr剩下的指标放右边(注意这里没有复共轭)，本张量的结构已知；如果收缩之后的张量结构未知，请调用构造函数。参数 cttype 可以为 'S', 'D' 和 'I', 具体用法参考相应的构造函数.

3.2.7 template<typename TSR, typename = typename std::enable_if<std::is_base_of<BaseTensor<DTTYP
TSR>::value>::type> BLASINT leftSVD(const char &thistype, const uint32_t &index, const char &Utype, Vector<RDTTYPE> &lambda, TSR &U, void* pool, const BLASINT &poolsize) ==> 将指标index移到最左边得到T，然后做奇异值分解 T = U * lambda * VD，这个函数的主要目的是得到U和lambda, VD是副产品。[reviewed?]

1. const char &thistype：'O' ==> VD 会存放到本张量中并且本张量的结构会发生改变(overwritten); 'K' ==>不存且本张量不会发生任何变化(kept); 'D'==> 本张量的数据被破坏(destroyed)
2. const BLASINT &index：需要分解的指标
3. const char &Utype：SVD分解得到U的方式，如果我们要求U是幺正矩阵，则type=='A'，否则 type=='S'
4. Vector<RDTTYPE>* lambda：奇异值
5. TSR* U：SVD分解得到的幺正矩阵
6. void* pool：辅助空间
7. const BLASINT &poolsize：pool的大小，in sizeof(byte).

3.2.8 BLASINT rightSVD(const char &thistype, const BLASINT &index, const char &type, double* lambda, const BLASINT &lmbdsize, DATATYPE* VD, const BLASINT &VDsize, DATATYPE* tmpdata, const BLASINT &tdsize, DATATYPE* work, const BLASINT &wksize, double* rwork, const BLASINT &rwksize) ==> 将指标移到最右边得到T，并做奇异值分解 T = U * lambda * VD, 这个函数主要目的是得到VD，和 lambda，U是副产品。[reviewed?]

3.2.9 void highOrderSVD(const BLASINT &num, const BLASINT* index, DATATYPE**U, const BLASINT* Usize, DATATYPE* lambdamax, const BLASINT &lmbdsize, const BLASINT* dimtrlbd, DATATYPE* tmpdata, const BLASINT &tdsize, DATATYPE* work, const BLASINT &wksize, double* rwork, const BLASINT &rwksize) ==> 高阶奇异值分解，例如：$T_{...i...j...k...l...} = U_{i\alpha}U_{j\beta}U_{k\gamma}U_{l\delta}S_{...\alpha...\beta...\gamma...\delta...}$,所有的U要求是幺正矩阵，分解完以后，core tensor S 存放在本张量并不做截断 [reviewed?]

1. const BLASINT &num：做高阶奇异值分解的腿的数目

2. **const BLASINT* index**：腿的指标；先将需要分解的腿移到最左边，然后做奇异值分解
3. **DATATYPE** U**：分解得到的矩阵，这里要求这些矩阵都是幺正矩阵
4. **DATATYPE* lambdamax**：存放奇异值分解的lambda，维数为所有需要分解的腿的维数之和，奇异值按照分解的腿的顺序依次存放
5. **const BLASINT* dimtrlbd**：每个分解的腿中，需要保留的奇异值的数目；如果超过了实际的奇异值的数目，对应的值就是0
6. **DATATYPE* tmpdata**：临时数组，和本地张量的数据空间一样大
7. **DATATYPE* work**：临时数组
8. **const BLASINT &wksize**：size of work
9. **double* rwork**：临时数组（DATATYPE=Complex）；没有用(DATATYPE=double)

3.2.10 void highOrderSVD(const BLASINT &num, const BLASINT* index, const BLASINT* dimtru, DATATYPE**U, double* lambdamax, const BLASINT* dimtrlbd, DATATYPE* tmpdata, DATATYPE* work, const BLASINT &lwork, double* rwork) ==> 高阶奇异值分解，所有的U都是幺正矩阵，core tensor S 相应的指标根据dimtr做了截断，注意本张量的维数发生了变化，数据的存储空间依旧是原来的存储空间[reviewed?]

1. **const BLASINT &num**：奇异值分解的腿的数目
2. **const BLASINT* index**：奇异值分解的腿的指标
3. **const BLASINT* dimtru**：
4. **DATATYPE* U**：
5. **double* lambdamax**：存放奇异值，只保留截断之后的，数组的大小为 dimtrlbd[0]+dimtrlbd[1]+...+max(index[i])

3.2.11 void highOrderSVD(const BLASINT &num, const BLASINT* index, const BLASINT* dimtru, DATATYPE**U, double* lambdamax, const BLASINT* dimtrlbd, DATATYPE* tmpdata, DATATYPE* work, const BLASINT &lwork, double* rwork, BLASINT &iwork) ==> 高阶奇异值分解，U根据保留奇异值的数目做了截断，本张量的维数也发生了变化，数据空间存放截断后的core tensor。奇异值分解调用lapack中gesvdx(...)完成。

3.2.12 void rankCombination(const BLASINT &rstart, const BLASINT &rnum)

3.2.13 void rankDecomposition(const BLASINT &rindex, const BLASINT &rnum, const BLASINT* rdim)

3.2.14 bool reshape(initializer_list<BLASINT> rd)