Group Meeting

xz

2015-07

Content

- Distributed ML at Douban
  - Stories
  - Lessons
  - Advice
- C++ Tips
  - C++ Skills
  - C++ Tricks
  - Advice
- Future Interest

Distributed ML at Douban

Story

Lesson

Advice

pSVD

- 2012 Summer Intern
- SVD every half a week, can we speed up?
- Solution1: impl matrix factorization(easy)
- **Solution2: pSVD(challenging, hot research)**

pSVD

- Unexpected C++(C++11 exactly)
- Search out a survey
- Choose the existing version in libdoubanm to parallelize
- One month of coding, one month of debuging
- 8 workers with 2.x speedup
- Memory do not scale

Algorithm 6 (Thick-restart Lanczos Bidiagonalization)

Input: Matrix A, initial vector q_1, and number of steps k
Output: `nev` Ritz triplets, with `nev` $\leq \ell < k$

Normalize q_1
Set $\ell = 0$
Restart loop
Run one-sided Lanczos bidiagonalization (Algorithm 7)
Compute the SVD of $B_k = X_k \Sigma_k Y_k^*$
Compute residual norm estimates, $|\rho_i|$, with $\rho_i = \beta_k e_k^* x_i$
Exit if enough converged singular triplets, otherwise lock newly converged triplets
Update ℓ and set $q_{\ell+1} \leftarrow q_{k+1}$
Compute Ritz vectors, $Q_\ell \leftarrow Q_k Y_{:,1:\ell}$, $P_\ell \leftarrow P_k X_{:,1:\ell}$

Insert ρ_i in the appropriate positions of B_k
end

Algorithm 7 (One-Sided Lanczos Bidiag. – restarted, with enhancements)

$p_{\ell+1} = Aq_{\ell+1}$
For $i = 1, 2, \ldots, \ell$
    $p_{\ell+1} = p_{\ell+1} - B_{i,\ell+1} p_i$
end
For $j = \ell+1, \ell+2, \ldots, k$
    $q_{j+1} = A^* p_j$
    $c = Q_j^* q_{j+1}$
    $\rho = \|q_{j+1}\|_2$
    $\alpha_j = \|p_j\|_2$
    $p_j = p_j / \alpha_j$
    $q_{j+1} = q_{j+1} / \alpha_j$
    $c = c / \alpha_j$
    $\rho = \rho / \alpha_j$
    $q_{j+1} = q_{j+1} - Q_j c$
    $\beta_j = \sqrt{\rho^2 - \sum_{i=1}^{j} c_i^2}$
    If $\beta_j < \eta\rho$
        $c = Q_j^* q_{j+1}$
        $\rho = \|q_{j+1}\|_2$
        $q_{j+1} = q_{j+1} - Q_j c$
        $\beta_j = \sqrt{\rho^2 - \sum_{i=1}^{j} c_i^2}$
    end
    $q_{j+1} = q_{j+1} / \beta_j$
    If $j < k$
        $p_{j+1} = Aq_{j+1} - \beta_j p_j$
    end
end

Gain

- Meet ChangSheng
- Learn MPI/C++11/libdoubanm
- Learn parallel computing

Lessons

Developing C++ is inefficient?

How to debug C++/C++11/libdoubanm code?

Can memory be scalable?

Advice

Capability of each worker is different

Advice

Choose simple alg to parallelize as possible as you can

Reference

- Large scale Singular Value Decomposition and applications in Machine Learning
- Restarted Lanczos Bidiagonalization for the SVD in SLEPc
- SLEPc
- PETSc

SVD in Paracel

- A novel approach
  - Step1: matrix factorization
  - Step2: QR method
  - Complexity
  - Partition
- Whiteboard…

Followup

“虽Google早就有了MapReduce等并行计算工具，但由于SVD很难拆成不相关子运算，即使在Google内部以前也是无法利用并行计算的优势来分解矩阵。直到2007年，Google中国（谷歌）的张智威博士带领几个中国的工程师以及实习生实现了SVD的并行算法，这是Google中国对世界的一个贡献。”

——《数学之美》

Parasol

- 2013 Summer
- Python prototype, advice from beansdb
- Impled basic paradigms(about 4weeks)
- Poor performance, optimize with Cython
- Not easy to use/develop

Gain

Parallel vs Distributed

Quantitative vs Qualitative

Think Mapreduce/Spark/Pregel

Basic probs of distributed computing

Lesson

System hierarchy and abstract is important

Lesson

Unit test is important and time saving

Lesson

C plus plus is different from Python

Advice

Low level modules must be abstract enough

Interface muse be replaceable

Advice

Use same languages to write a prototype

Roraima

- 2013, Winter
- Factor based recommendation
- Study LSH, try to deduce with factor model
- Not possible(a paper)
- Curse of dimensionality
- Impl Balltree
- Not that optimized with real data

Gain

- Understanding different between dp/cosine
- Sense of accomplishment from impl a paper

Lesson

Paper may be misleading and useless

Advice

Baseline is the very first thing to do

Reference

P. Ram & A. Gray, Maximum Inner-Product Search using Tree Data-Structures, 2012

Maximum Inner-Product Search Using Cone-Trees,KDD12

Prepermutation

- 2014, Winter
- Factor based recommendation is bad in book/movie data
- Prepermutate the input rating matrix
- Netflix data in 4,5 hours, memory expensive
- FM data?

Google Chrome 星期日 16:49 zh wuh

Password Request - xunzl

https://mail.google.com/mail/u/0/#search/zhangyf07%40gmail.com/149748f9b712d411

cs.brown.edu | online-computation | 2014 | heuristic | TecBlog | index.html | box | xunzhang | Wolfram Alpha | stackoverflow | Bitbucket | arXiv.org | 其他

系统已不再支持此Chrome版本。 请升级至 支持的浏览器。 关闭

Google zhangyf07@gmail.com hong

Gmail 移至收件箱 更多操作 第1封，共1封

写邮件

收件箱
已加星标
重要邮件
已发邮件
草稿 (3)
圈子

hong
Ruizhe Li
Tian Pan
Changsheng Jiang
Huimin Lee
Jinglei Ren

Yongfeng Zhang <zhangyf07@gmail.com> 14/11/4

发送至 我

Hi 吴师兄：

我发现造成计算时间很长的原因了，给你带来了不便非常抱歉！

问题出在 libmetis/bmetis.c这个小函数上，这个函数的作用是对于当前给定的一个子矩阵（通过指定原始矩阵中的某些行和某些列的方式给出，这些行和列的交叉点就是这个子矩阵），计算它里面的非零值的个数，在公布代码之前，我是同随机采样的方法估计非零值的个数，也就是在这个子矩阵中随机采样一定数量的点，看有多少是非零的，然后估计出密度，再根据面积反推出非零值的个数，主要是为了提速；在公布代码的时候为了统计精确的非零值个数，我把这部分直接变成loop search了，但是没有考虑到复杂度的问题，于是这个小函数的复杂度是O(nrows*ncols)，由于这个小函数会被频繁调用，因此对于大数据非常耗时。如果这部分复杂度可以忽略不计，其它部分都是很快的，也不需要并行。

我找时间在仔细改一下这个部分，简单地可以恢复为采样统计的方法，也可以改成对整个大矩阵中的所有非零值都loop一遍count到相应的子矩阵中，而不需要每次都对小矩阵统计。当时只把这部分简单改了一下没有仔细测试就发布了，非常抱歉，幸好师兄发现和指出！

祝好！
张永锋

hong wu <xunzhangthu@gmail.com> 14/11/4

发送至 Yongfeng

ok。我们在豆瓣fm上factor model很成功，但在其他产品线的数据上发现了一些关键问题（并不是性能问题），正在尝试解决。你的工作是一种思路，因此想尝试一下。我没有细看metis的code，本来想直接run看效果的，看来现在只能尝试按你的思路改改看了。

Yongfeng Zhang
添加到圈子
显示详细信息

Advice

Think twice before implementing a ML paper

Reference

- Localized Matrix Factorization for Recommendation based on Matrix Block Diagonal Forms(www13)

Spectral Clustering

- 2014, Autumn
- Probs from FM(must be distributed)
- User factor clustering
- pspectralclustering from google(not flexiable)
- Hard…
- One month math, one month coding….
- 80% finished, to be debuged and refactord...

Spectral Clustering

- Similarity graphs
  - Mutual k-nearest neighbor graph
- Graph laplacian matrixes

Algorithm

Normalized spectral clustering according to Shi and Malik (2000)

Input: Similarity matrix $S \in \mathbb{R}^{n \times n}$, number k of clusters to construct.

- Construct a similarity graph by one of the ways described in Section 2. Let W be its weighted adjacency matrix.
- Compute the unnormalized Laplacian L.
- **Compute the first k generalized eigenvectors $u_1, \ldots, u_k$ of the generalized eigenproblem $Lu = \lambda Du$.**
- Let $U \in \mathbb{R}^{n \times k}$ be the matrix containing the vectors $u_1, \ldots, u_k$ as columns.
- For $i = 1, \ldots, n$, let $y_i \in \mathbb{R}^k$ be the vector corresponding to the i-th row of U.
- Cluster the points $(y_i)_{i=1,\ldots,n}$ in $\mathbb{R}^k$ with the k-means algorithm into clusters $C_1, \ldots, C_k$.

Output: Clusters $A_1, \ldots, A_k$ with $A_i = \{j | \, y_j \in C_i\}$.

- Solve Eigen Vectors distributed
- Combing SVD
- Whiteboard….

Lesson

Run existing tools

Not just read docs

Advice

Disk I/O may be useful to simplify(mapreduce)

Open Sourcing Rethink

- 2015, Spring
- Usability is important
- Document is important
- How to engage users worldwide?
- Is fault tolerant necessary at Douban?

Other Advice

- Coding with general data format
- Write a sequential version at first
- Use existing frameworks(dpark/doubanm/paracel)

Discussion

- Internet service
- Search, Ads
  - 100% algorithms
- Recommendation
  - 40% algorithms + 60 products
  - Gap between RMSE(model) & performance(bad/good case)
- Algorithm at Douban
  - **60% model abstract**
  - 30% engineering
  - 10% research

C++ Tricks

Basic

- Dynamic Interface

  DouCoder -> Coder

  Coder *xz = new DouCoder(…)

  Coder *gq = new Coder(…)

- Static Interface
  - Type overload
  - Concept overload

Interface Design(OOP)

```
struct shape {
   virtual void draw(canvas *c) const = 0;
   virtual double area() const = 0;
   inline virtual ~shape() {}
};

struct rectangle {
   double x,y, w, h;
   inline virtual void draw(canvas *c) const {…}
   inline virtual double area() const { return w * h; }
};

struct circle {
   double x,y, r;
   inline virtual void draw(canvas *c) const {…}
   inline virtual double area() const { return M_PI * r * r; }
};
```

Interface Design(OOP)

```
struct drawable {
   virtual void draw(canvas *c) const = 0;
   inline virtual ~drawable() {}
};

struct with_area {
   virtual double area() const = 0;
   inline virtual ~with_area() {}
};

struct shape: drawable, with_area {
   inline virtual ~shape() {}
};

struct rectangle: shape, … {
   inline virtual double area() const {…}
   inline virtual void draw(canvas *c) const {…}
};
```

Interface Design(static)

```
template <class T>
Enable_if<has_method_draw<T>::value>
draw(const T & d, canvas *c) {
   d.draw(c);
}

template <class T>
Enable_if<has_method_draw<T>::value>
area (const T & d) {
   d.area();
}
```

Interface Design(static)

```
struct OtherRectangle {
   double acreage() const;
   void canvas_draw(canvas *c) const;
};

double area(const OtherRectangle & d) {
   return d.acreage();
}

void draw(const OtherRectangle & d, canvas *c) {
   d.canvas_draw(c);
}
```

C++ Tricks

What is

C++ Tricks

- Rvalue References(&&)
  - Std::move
  - Std::forward
  - Move constructor
  - Move assignments
  - Std::swap(compiler code)

C++ Tricks

- Meta Programming
  - Template specialization
  - SFINAE(Substitution Failure is Not An Error)
    - std::enable_if

C++ Tricks

- Type Erasure
  - std::function
  - boost::any
  - douban::opt::problem

C++ Tricks(optional)

- CRTP(Curiously Recurring Template Pattern)
  - douban::mat_proxy
  - douban::vec_proxy

C++ Tricks(optional)

- Variadic Tempaltes
  - paracel::kvclt::paste

C++ Tricks(optional)

- Powerful Macro System
  - ##
  - __VA_ARGS_
  - glog
  - douban::clog

C++ Tricks

- Connect with Python
  - CAPI
  - boost::python
  - **Cython**
  - doubanm/pyu
- Connect with any other PLs
  - thrift
  - douban::io_port
    - douban::doubandb_port

std::future

```
paracel::async_functor_type future;
paracel_update(k, v, update_fn, func, future);
// do some computation
bool r = paracel::wait(future);
if(r) {
    // continue iteration
}
```

Common Mistakes

- Std::thread
  - No start function
- Std::getline
  - No newline in the end
- Function inside a class
  - this pointer ahead
- Memory Leakage
  - std::unique_ptr

Pros

- Memory
  - C/C++: sizeof(int) = 4
  - Python
    - sizeof(int)
    - + sizeof(pointer to type)
    - + sizeof(reference counter)
    - + padding
    - + dynamic memory management
    - In[1]: sys.getsizeof(123)
    - Out[1]: 24

Pros

- CPU
  - C int plus(mov, add)
  - Python int plus
    - a + b
    - look up ‘a’ and ‘b’ in locals() / globals() if not ready
    - check a->type and a->type->__add__
    - call a->type->plus(a, b)
    - get a->value
    - check b’s type
    - if b->type != number, convert to number
    - a->value + number(v)

Cons

- No Free Lunch
  - Poor develop efficiency
    - No social
  - Poor compile experience
    - No friends
- Debug is hard
  - C++11 is an example

Advice

Read Source Code

I. usage

II. interface

III. implementation

Advice

Do not try to be a language lawyer

Advice

Learn Different Views of the World

Procedural: C/Fortran/COBOL/Pascal

OO(compiled/dynamic): C++/Java/Python/Go/Scala..

Declarative: Haskell/ML/Prolog

functional: Haskell/Erlang/Ocaml

Lisp…

Web: JavaScript, HTML, CSS

Data: SQL

Advice

Depends on Project?

Depends on Company?

Future Interest

- Breadth first search for large Natural Graphs
- Adjust Topk recommendation
- Computer vision
- Fault tolerance
- Debug tools for distributed system

Trip to Paracel

Thank You