小政哥
码龄2年 暂无认证

21	116	66	60	7万+
原创	粉丝	获赞	评论	访问
1003	156	14万+	8万+	
积分	收藏	周排名	总排名	等级

TA的主页 私信 关注

搜博主文章

最新文章

- vtkImageData转换为cv::Mat
- suitesparse-metis-for-windows Release 1.3.1 CMAKE配置文件
- 我是如何五个月从小白到AI工程师?
- 到底什么是非线性规划?
- Ceres Solver 官方教程学习笔记（十三）——非线性最小二乘问题的解算Solving Non-linear Least Squares （上）

分类专栏

学习笔记	22篇
Issues	2篇
安装教程	2篇
转载	3篇

归档

2018

6月	5月	4月	3月
1篇	2篇	11篇	10篇

热门文章

- 到底什么是非线性规划? 13868
- Ceres Solver 在Windows下安装配置笔记 7592
- Ceres Solver 官方教程学习笔记（一） 7446
- Ceres Solver 官方教程学习笔记（五）——光束法平差Bundle Adjustment 4485
- VTK相机类vtkCamera原理及用法 4259

最新评论

Ceres Solver 在Win...
cuijixiang000：编译成功，特来感谢！！...

Ceres Solver 官方教程...
Winkin_Chan：[reply]sinat_28752257[/reply]struct Ftotal { template<typename T> ...

Ceres Solver 在Win...
eric6079hn：[reply]xiamaomao28[/reply]你的Eigen3 install是不是不在C:/Program ...

Ceres Solver 在Win...
baidu_38505667：我的ceres release 模式生成完了 切换debug时，没有见debug ...

Ceres Solver 官方教程...
lianghu3124：[reply]qq_36437317[/reply]你好，请问你弄明白了吗？可否解答一

目录

Ceres Solver 官方教程学习笔记（九）——自动微分法Automatic Derivatives

原创 小政哥 2018-04-05 19:57:38 1508 收藏 3 版权
分类专栏： 学习笔记

> 这篇文章翻译自官方教程Automatic Derivatives并且参考了少年的此间的博客文章Ceres-Solver学习笔记(5)

现在我们将讨论自动微分算法。它是一种可以快速计算精确导数的算法，同时用户只要做与数值微分法类似的工作。下面的代码片段实现了对Rat43（见前两节）的`CostFunction`。

```
struct Rat43CostFunctor {
  Rat43CostFunctor(const double x, const double y) : x_(x), y_(y) {}

  template <typename T>
  bool operator()(const T* parameters, T* residuals) const {//变化1
    const T b1 = parameters[0];
    const T b2 = parameters[1];
    const T b3 = parameters[2];
    const T b4 = parameters[3];
    residuals[0] = b1 * pow(1.0 + exp(b2 -  b3 * x_), -1.0 / b4) - y_;
    return true;
  }

  private:
    const double x_;
    const double y_;
};


CostFunction* cost_function =
      new AutoDiffCostFunction<Rat43CostFunctor, 1, 4>(    //变化2
        new Rat43CostFunctor(x, y));
```

复制

我把对应的数值微分法代码贴在这里以供对比。

```
struct Rat43CostFunctor {
  Rat43CostFunctor(const double x, const double y) : x_(x), y_(y) {}

  bool operator()(const double* parameters, double* residuals) const {
    const double b1 = parameters[0];
    const double b2 = parameters[1];
    const double b3 = parameters[2];
    const double b4 = parameters[3];
    residuals[0] = b1 * pow(1.0 + exp(b2 -  b3 * x_), -1.0 / b4) - y_;
    return true;
  }

  const double x_;
  const double y_;
}

CostFunction* cost_function =
  new NumericDiffCostFunction<Rat43CostFunctor, FORWARD, 1, 4>(
    new Rat43CostFunctor(x, y));
```

注意，与数值微分法相比，在定义自动微分的Functor时，唯一的区别是对操作符`operator()`的设置。
在数值微差的情况下

```
//数值微分法
bool operator()(const double* parameters, double* residuals) const;

//自动微分法
template <typename T> bool operator()(const T* parameters, T* residuals) const;
```

这个变化有什么影响呢？下表比较了使用各种方法对Rat43进行计算残差和雅可比矩阵的时间。

CostFunction	Time (ns)
Rat43Analytic	255
Rat43AnalyticOptimized	92
Rat43NumericDiffForward	262
Rat43NumericDiffCentral	517
Rat43NumericDiffRidders	3760
Rat43AutomaticDiff	129

我们可以使用自动微分(Rat43AutomaticDiff)来得到精确的微分。而这与编写数字微分的代码量相差不多，但比优化后的解析微分法只慢40%。为了研究它的工作原理，必须要学习**二元数(Dual number)**和**射流(Jet)**

二元数(Dual number)和射流(Jet)

阅读这一小节和下一节关于实现Jets的内容，与在Ceres求解器中使用自动微分没有直接关系。但是，在调试和推理自动微分的性能时，了解Jets的工作原理是非常有用的。

二元数是实数的一个延伸，类似于复数。复数则通过引入虚数来增加实数，比如i，二元数引入了一个*极小(infinitesimal)*二元数单位，比如ϵ，且$\epsilon^2=0$（平方后太小可以忽略）。一个二元数$a+v\epsilon$包含两个分量，实分量a和极小分量的v。令人惊喜的是，这个简单的变化带来了一种方便的计算精确导数的方法，而不需要复杂的符号表达式。
例如，考虑函数

$$f(x)=x^2,$$

然后

$$\begin{aligned}f(10+\epsilon)&=(10+\epsilon)^2\\&=100+20\epsilon+\epsilon^2\\&=100+20\epsilon\end{aligned}$$

观察ϵ的系数，我们发现$Df(10)=20$。事实上，这个规律可以推广到不是多项式的函数。考虑一个任意可微函数$f(x)$。然后我们可以计算$f(x+\epsilon)$，通过在x附近做泰勒展开，这就得到了无穷级数

$$\begin{aligned}f(x+\epsilon)&=f(x)+Df(x)\epsilon+D^2f(x)\frac{\epsilon^2}{2}+D^3f(x)\frac{\epsilon^3}{6}+\cdots\\f(x+\epsilon)&=f(x)+Df(x)\epsilon\end{aligned}$$

记住，$\epsilon^2=0$。
射流Jet是一个n维二元数。我们定义n个极小单位ϵ_i，$i=1,\ldots,n$。并且存在性质$\forall i,j\ :\epsilon_i\epsilon_j=0$。射流数由实数$a$和$n$维极小分量组成。

$$x=a+\sum_j v_j\epsilon_j$$

为了简化我们改写为这种形式

$$x=a+\mathbf{v}.$$

然后，使用泰勒级数展开，我们可以看到：

$$f(a+\mathbf{v})=f(a)+Df(a)\mathbf{v}.$$

对多变量函数$f:\mathbb{R}^n\to\mathbb{R}^m$相似。对于自变量$x_i=a_i+\mathbf{v}_i$，$\forall i=1,\ldots,n$：

$$f(x_1,\ldots,x_n)=f(a_1,\ldots,a_n)+\sum_i D_if(a_1,\ldots,a_n)\mathbf{v}_i$$

如果每个选取的极小量$\mathbf{v}_i=e_i$是i^{th}标准基向量，那么上面的表达式就可以简化为

$$f(x_1,\ldots,x_n)=f(a_1,\ldots,a_n)+\sum_i D_if(a_1,\ldots,a_n)\epsilon_i$$

我们可以通过查找ϵ_i的系数来提取雅可比矩阵的坐标。

实现射流(Jet)

为了让上面学到的内容在实践中发挥作用，我们需要能够计算函数f的值，不仅在自变量是实数的时候，也需要在自变量是二元数的情况下适用。但是通常我们并非通过泰勒展开式来求函数值。这也就是为什么我们需要用到C++模板和操作符重载。下面的代码段实现了`Jet`类以及对该类的一些操作和函数。

```
template<int N> struct Jet {
  double a;
  Eigen::Matrix<double, 1, N> v;
};

template<int N> Jet<N> operator+(const Jet<N>& f, const Jet<N>& g) {
  return Jet<N>(f.a + g.a, f.v + g.v);
}

template<int N> Jet<N> operator-(const Jet<N>& f, const Jet<N>& g) {
  return Jet<N>(f.a - g.a, f.v - g.v);
}

template<int N> Jet<N> operator*(const Jet<N>& f, const Jet<N>& g) {
  return Jet<N>(f.a * g.a, f.a * g.v + f.v * g.a);
}

template<int N> Jet<N> operator/(const Jet<N>& f, const Jet<N>& g) {
  return Jet<N>(f.a / g.a, f.v / g.a - f.a * g.v / (g.a * g.a));
}

template <int N> Jet<N> exp(const Jet<N>& f) {
  return Jet<T, N>(exp(f.a), exp(f.a) * f.v);
}

// This is a simple implementation for illustration purposes, the
// actual implementation of pow requires careful handling of a number
// of corner cases.
template <int N>  Jet<N> pow(const Jet<N>& f, const Jet<N>& g) {
  return Jet<N>(pow(f.a, g.a),
                g.a * pow(f.a, g.a - 1.0) * f.v +
                pow(f.a, g.a) * log(f.a); * g.v);
}
```

有了这些重载的函数，我们现在可以用一个`Jets`数组来调用`Rat43CostFunctor`（见Ceres Solver 官方教程学习笔记（八）——数值微分法Numeric derivatives），而不是`double`双精度类型。将其与初始化的`Jets`结合起来，我们就可以计算雅可比矩阵了：

```
class Rat43Automatic : public ceres::SizedCostFunction<1,4> {
 public:
  Rat43Automatic(const Rat43CostFunctor* functor) : functor_(functor) {}
  virtual ~Rat43Automatic() {}
  virtual bool Evaluate(double const* const* parameters,
                        double* residuals,
                        double** jacobians) const {
    // Just evaluate the residuals if Jacobians are not required.
    if (!jacobians) return (*functor_)(parameters[0], residuals);

    // 初始化Jets，四个待求参数
    ceres::Jet<4> jets[4];
    for (int i = 0; i < 4; ++i) {
      jets[i].a = parameters[0][i];
      jets[i].v.setZero();
      jets[i].v[i] = 1.0;
    }

    ceres::Jet<4> result;
    (*functor_)(jets, &result);

    // 把Jet的值（前面提到的，极小单位分量的系数）复制出啦.
    residuals[0] = result.a;
    for (int i = 0; i < 4; ++i) {
      jacobians[0][i] = result.v[i];
    }
    return true;
  }

 private:
  std::unique_ptr<const Rat43CostFunctor> functor_;
};
```

这就是`AutoDiffCostFunction`的核心工作原理。

陷阱

自动微分使用户不必计算和推理Jacobians的符号表达式，但是这个捷径是有代价的。例如，考虑以下简单的函数：

```
struct Functor {
  template <typename T> bool operator()(const T* x, T* residual) const {
    residual[0] = 1.0 - sqrt(x[0] * x[0] + x[1] * x[1]);
    return true;
  }
};
```

查看计算残差的代码，没有人预见到任何问题。但是，如果我们看一下雅可比矩阵的解析表达式

$$\begin{aligned}y&=1-\sqrt{x_0^2+x_1^2}\\D_1y&=-\frac{x_0}{\sqrt{x_0^2+x_1^2}},\ D_2y=-\frac{x_1}{\sqrt{x_0^2+x_1^2}}\end{aligned}$$

我们发现它在$x_0=0$，$x_1=0$处是不确定的。

这个问题没有完美的解决方案。在某些情况下，我们需要明确地指出可能出现的不确定的点，并使用使用L'Hopital's rule"的替代表达式（例如参见rotation.h中的一些转换例程），在其他情况下，可能需要对表达式进行正则化，以消除这些点。

Ceres Solver 官方教程学习笔记（X）——自动微分法接口Interfacing with Automatic Differentiation 小政哥的私房博客 889
这篇文章翻译自官方教程Automatic Derivatives并且参考了少年的此间的博客文章Ceres-Solver学习笔记(5)在成本函数的有一个显式表达式的情况下，自...

Ceres Solver 官方教程学习笔记（六）——关于微分计算On Derivatives 小政哥的私房博客 991
本节主要根据也Ceres官方教程On Derivatives翻译而成与所有基于梯度的优化算法一样，Ceres Solver也是基于评估域中任意点的目标函数及其导数。事...

优质评论可以帮助作者获得更高权重 抢沙发 评论

非线性优化库Ceres学习笔记2：自动求导 JunJun 382
Hello World!考虑下面的最优化问题：1 编写一个代价函数struct CostFunctor { template <typename T> bool operator()(const T* const x, T* residual) const...

自动微分(Automatic Differentiation)简介 xiaomeng29的博客 410
转载地址：https://blog.csdn.net/aws3217150/article/details/70214422上面的文章写的非常好，不多做介绍，只说下自己对于两种自动微分的理解。1. 首...

Automatic Derivatives u012314784的博客 120
Automatic Derivatives我们现在将要讨论自动求导。这个技术可以快速地计算准确的导数，当用户需要使用数值求导时，自动导数可以提供相同的效果。不...

Ceres自动求导（AutomaticDiff）初探 LeatherWang的博客 2487
g2o中如果没有定义这个边的linearizeOplus()，就会调用数值求导。但是g2o的数值求导比较慢，效果较差。所以下面探讨在g2o中嵌入ceres的自动求导，...

Ceres Solver 官方教程学习笔记（八）——数值微分法Numeric derivatives 小政哥的私房博客 1054
这篇文章翻译自官方教程Numeric derivatives并且参考了少年的此间的博客文章Ceres-Solver学习笔记(5)利用analytic derivatives的另一个极端形式是 num...

Ceres Solver 官方教程学习笔记（一） 小政哥的私房博客 7469
本文主要基于Ceres Solver 官方Tutourial选取要点进行翻译，并适当补充外部内容对某些概念进行简要注释。主要用于个人学习备忘之作用。简介Ceres可...

一文读懂自动微分（ AutoDiff）原理 南瓜派三蔬 1420
整理自《Hands On ML——Appendix D》，有改动，如有不解请参考原文（感觉原文也多少有点问题）。1. Manual Differentiation2. Symbolic Differentiati...

TensorFlow自动求导原理 qq_25094489的博客 5867
原理：TensorFlow使用的求导方法称为自动微分（AutomaticDifferentiation），它既不是符号求导也不是数值求导，而类似于将两者结合的产物。最基本的...

Ceres-Solver库使用（三）--导数(Derivatives) yizhou2010的专栏 4476
ceres-solver类似于大多数的优化包一样，依赖于能够计算目标函数在任意参数值上的每项函数值和每项导数值，这些计算值的准确性和高效性是获得好的...

Ceres Solver 官方教程学习笔记 SLAM算法攻城狮 234
https://blog.csdn.net/wzheng92/article/category/7496047https://blog.csdn.net/cqrtxwd/article/details/78956227

毕业5年，我问遍了身边的大佬，总结了他们的学习方法 敖丙 32万+
我问了身边10个大佬，总结了他们的学习方法，原来成功都是有迹可循的。

ceres优化库的一点总结（问题构建总结和求导方式的分析） supengufo的博客 293
1.Ceres中求解一个优化问题的结构背景：在SLAM中，很多问题都是在求解Translation（包含旋转和平移量），因此这里以其为代表，来分析使用ceres如...

Ceres Solver使用 chengwei0019 2106
非线性最小二乘Ceres可以求解以下形式的有界约束非线性最小二乘问题：minx12∑ipi(//fi(xi1,⋯,xik)//2)s.t.lj≤xj≤ujminx12∑ipi(∥fi(xi1,⋯,xik)∥2)s.t.lj≤xj≤uj这...

非线性优化Ceres手动求导数值求导解析求导使用示例 沧海一帆的专栏 937
参考Ceres官方文档http://www.ceres-solver.org/nnls_tutorial.html#hello-world实现手动求导进行非线性优化。// Author: keir@google.com (Keir Mierle)//// ...

Ceres Solver使用总结 MyArrow的专栏 5055
1.基本概念CeresSolver是一个开源C++库，用于建模和解决大型复杂的优化问题。用途具有边界约束的非线性最小二乘问题minx12∑ipi(//fi(xi1,...,xij)//2)\m...

Ceres使用（二） u013341645的博客 1010
非线性最小二乘IntroductionCeres可以解决这种形式的边界约束鲁棒非线性最小二乘问题： minx12∑ipi(///fi(xi1,...,xik)///2)s.t.lj≤xj≤uj(1)\begin{split}\min_{...

Python可以这样学（第一季：Python内功修炼） 02-23
董付国系列教材《Python程序设计基础》、《Python程序设计（第2版）》、《Python可以这样学》配套视频，讲解Python 3.5.x和3.6.x语法、内置对象用...

获取Linux下Ftp目录树并逐步绑定到treeview 06-08
在linux下抓取目录树，双击后获取该节点子节点（逐步生成）。另外有两个类，一个是windows下的（一次性获取目录树），一个是linux下的（足部获取...