SciStaEBD Documentation

ElegantLaTeX 经典之作

Author: ZhouYao

Institute: ElegantLaTeX Program

Date: 2020 年 2 月 9 日

Version: 0.8

ElegantLaTeX Program

Victory won't come to us unless we go to it. — M. Moore

目录

Appendix 1 Getting Started

SciStaEBD is a header only library and it has already contains a copy of another two header-only libraries: boost.math and Eigen3. First download at SciStaEBD on GitHub and then copy the internal folder ”SciStaEBD” to your projects or complier’s include directory.

Appendix 2 Design Philosophy

2.1 Goals

2.2 Principles

2.3 Programming Convention

2.3.1 Template Arguments

Argument	Meaning
`S`	Scalar type such as `double`.
`V`	Eigen vector type such as `VectorXd`.
`M`	Eigen matrix type such as `MatrixXd`.
`SV`	Scalar or Vector.
`XSV, YSV`	`X` and `Y` mean input and output respectively.

Appendix 3 Mathematic Core Functions(MathCore)

3.1 logistic() and logit()

Formula:

$$logistic(x, x0, L, k) = \frac{L}{1 + e^{-k(x-x0)}}$$

$$logistic(x) = \frac{1}{1 + e^{-x}}$$

$$logit(x) = \ln(\frac{x}{1 - x})$$

Defination:

```
template<typename T>
inline T logistic(T x, T x0 , T L = 1, T k = 1);

inline double logistic(double x);

inline double logit(double x);
```

3.2 softmax() and sigmoid()

Formula:

$$softmax(\mathbf{x}) = \frac{e^{x_i}}{\sum e^{x_k}}$$

$$sigmoid(x_i) = \frac{1}{1 + e^{-x_i}}$$

One can see that sigmoid function is simply element-wise logistic function.

Defination:

```
template<typename EigenV>
inline EigenV softmax(EigenV x);

template<typename EigenV>
inline EigenV sigmoid(EigenV x);
```

Appendix 4 Statistics Core Functions(StatCore)

Appendix 5 Optimization and Equation System Solving(Solve)

5.1 Usage

5.1.1 Optimization

5.1.1.1 Variable Types

1. x, y, gradient are forced to be `VectorXd` type and hessian are forced to be `MatrixXd` type. If x,y of objective function or nonlinear constraints are scalars, we still use a length one `VectorXd` to store it. One advantage of this is that we don’t have to deal with number of variables. Note that `VectorXd` is a column vector.

   If y is a vector of length more than one. One can set `SolveOption.type` to value either of ”least square” or ”norm”. For ”norm”, we optimize norm of y.
2. One can pass external data by pointer using template.

5.1.1.2 MATLAB style API

5.1.1.3 C++ Style Minimal Example

```
//Objective function.
double fun1(VectorXd x){
  return pow(x[0],2)+pow(x[1],2);
};

//Configure solver.
OptProblem<> problem("min",2);
problem.set_objective(fun1);
VectorXd x0;
x0<<10,10;
problem.set_x0(x0);
OptResult res=problem.solve();
cout<<res.x<<endl;
```

5.1.1.4 Constrained Optimization

5.1.1.5 User Supplied Gradient Function

5.1.2 Nonlinear System

5.2 Common Objects

5.2.1 SolveOption

5.2.2 SolveResult

5.3 Solve optimization problems

5.3.1 Choose A Solver

If a solver requires gradient and hessian, however they are not provided, then default difference approximation will be used.

- Special: `QPSolver`(Constrained QP)
- Unconstrained:
- Linear inequality constrained:`LCOBYQASolver`(doesn't use gradient and hessian).
- Nonlinear constrained:`LSSQPSolver`(Large scale SQP).
- Nonlinear least square:
- Evolutionary:`DESolver`.
- Heuristic Method: `PSOSolver`.

5.3.2 SolverBase

5.3.3 QPSolve

`QPSolve` is a basic component of many other solvers. It's not a derived class of `SolverBase`. It solve a problem:

$$\min_x f(x) = \frac{1}{2}x^T Gx + x^T c \tag{5.1}$$

$$subject Ax \leq b \tag{5.2}$$

Usage:

5.4 EQSystem - Solve nonlinear system of equations

Appendix 6 Econometrics Models(Econ)

Appendix 7 Non-parametric statistic models(NParmStat)

Appendix 8 Utility

8.1 EigenHelper

8.1.1 slice_by_set - indexing a matrix by integer set

Giving a matrix `mat` and a integer set `is`, dimension indicator `dim`. Return a matrix so that each column or row comes from `mat` indicated by `is`.

`dim=0` for select rows. `dim=1` for select columns.

Definition:

```
template<typename SV, typename IDXT>
SV slice_by_set(SV mat, IDXT is, int dim=0);
```

`IDXT` is the type of set. It can be `set<int>` or `vector<int>`. It's very useful when you want to manipulate a subset of rows or columns.

Example:

```
set<int> s;
s.emplace(1);
s.emplace(2);
s.emplace(0);

MatrixXd m(5, 4);
m.setRandom();

cout << slice_by_set(m,s) << endl;
cout << slice_by_set(m, vector<int>{ 0,1,2 }) << endl;
```

8.2 FunctionCollection

8.2.1 print_stl - print values in a STL container

Definition:

```
template<typename T>
void print_stl(T& v);
```

8.2.2 which - get indices of a value in a container

Definition:

```
template<typename V, typename S>
vector<int> which(const V& v, const S& val);
```

8.2.3 set2vec - convert a set to vector

Definition:

```
template<typename T>
vector<T> set2vec(const set<T>& s);
```