minFunc

Mark Schmidt (2005)

minFunc is a Matlab function for unconstrained optimization of differentiable real-valued multivariate functions using line-search methods. It uses an interface very similar to the Matlab Optimization Toolbox function *fminunc*, and can be called as a replacement for this function. On many problems, *minFunc* requires fewer function evaluations to converge than *fminunc* (or *minimize.m*). Further it can optimize problems with a much larger number of variables (*fminunc* is restricted to several thousand variables), and uses a line search that is robust to several common function pathologies.

The default parameters of *minFunc* call a quasi-Newton strategy, where limited-memory BFGS updates with Shanno-Phua scaling are used in computing the step direction, and a bracketing line-search for a point satisfying the strong Wolfe conditions is used to compute the step direction. In the line search, (safeguarded) cubic interpolation is used to generate trial values, and the method switches to an Armijo back-tracking line search on iterations where the objective function enters a region where the parameters do not produce a real valued output (ie. complex, NaN, or Inf).

Features

Some highlights of the non-default features present in minFunc:

- Step directions can be computed based on: Exact Newton (requires user-supplied Hessian), full quasi-Newton approximation (uses a dense Hessian approximation), limited-memory BFGS (uses a low-rank Hessian approximation - default), (preconditioned) Hessian-free Newton (uses Hessian-vector products), (preconditioned) conjugate gradient (uses only previous step and a vector beta), Barzilai and Borwein (uses only previous step), or (cyclic) steepest descent.
- Step lengths can be computed based on either the (non-monotone) Armijo or Wolfe conditions, and trial values can be generated by either backtracking/bisection, or polynomial interpolation. Several strategies are available for selecting the initial trial value.
- Numerical differentiation and derivative checking are available, including an option for automatic differentiation using complex-step differentives (if the objective function code handles complex inputs).
- Most methods have user-modifiable parameters, such as the number of corrections to store for L-BFGS, modification options for Hessian matrices that are not positive-definite in the pure Newton method, choice of preconditioning and Hessian-vector product functions for the Hessian-free Newton method, choice of update method;scaling;preconditioning for the non-linear conjugate gradient method, the type of Hessian approximation to use in the quasi-Newton iteration, number of steps to look back for the non-monotone Armijo condition, the parameters of the line search algorithm, the parameters of the termination criteria, etc.

Usage

minFunc uses an interface very similar to Matlab's fminunc. If you currently call 'fminunc (@myFunc,x0,options,myFuncArg1,myFuncArg2)', then you can use minFunc instead by simply replacing 'fminunc' with 'minFunc'. Note that by default minFunc assumes that the gradient is supplied, unless the 'numDiff' option is set to 1. minFunc supports many of the same parameters as fminunc (but not all), and has many parameters that are not available for fminunc. 'help minFunc' will give a list of parameters and their explanation.

Example

'example_minFunc' gives an example of running the various limited-memory solvers in minFunc with default options. It requires rosenbrock.m (the 2D Rosenbrock "banana" function) from the minimize.m webpage (minimize.m is also included in the demo if it is found on the path). Running 'example_minFunc' should

produce the following output:

```
Result after 25 evaluations of limited-memory solvers on 2D rosenbrock:
--------------------------------------
x1 = 0.0000, x2 = 0.0000 (starting point)
x1 = 1.0000, x2 = 1.0000 (optimal solution)
--------------------------------------
x1 = 0.7909, x2 = 0.6224 (minimize.m by C. Rasmussen)
x1 = 0.3640, x2 = 0.1342 (minFunc with steepest descent)
x1 = 0.4974, x2 = 0.2452 (minFunc with cyclic steepest descent)
x1 = 0.8758, x2 = 0.7664 (minFunc with spectral gradient descent)
x1 = 0.5840, x2 = 0.3169 (minFunc with Hessian-free Newton)
x1 = 0.7478, x2 = 0.5559 (minFunc with preconditioned Hessian-free Newton)
x1 = 1.0009, x2 = 1.0010 (minFunc with conjugate gradient)
x1 = 0.8551, x2 = 0.7232 (minFunc with scaled conjugate gradient)
x1 = 0.8943, x2 = 0.7909 (minFunc with preconditioned conjugate gradient)
x1 = 1.0000, x2 = 1.0000 (minFunc with limited-memory BFGS - default)
--------------------------------------
```

A note on the above performance results: The default parameters of minFunc were tuned based on log-linear statistical models (not on standard optimization problems like Rosenbrock's function).

To illustrate the use of Hessian-vector products and preconditioning in the newest version of minFunc, running 'example_minFunc_LR' will show how to train a logistic regression classifier under various configurations of the Hessian-vector product and preconditioning function. Hessian-vector products for the scaled conjugate gradient method, and preconditioning for the preconditioned conjugate gradient method, work similarly.

I prepared a more extensive set of examples of using minFunc on the webpage Examples of using minFunc.

Download

The complete set of .m files (and optional mex files for Windows/Mac/Linux) for the current version of minFunc are available here

Constrained Optimization

Previous versions of minFunc included extra functions that solved certain types of constrainted optimization problems (such as optimization with lower and/or upper bounds on the variables). These functions are no longer included, for constrained optimization please see minConf. Alternately, Michael Zibulevsky has written a penalty/barrier multiplier method for large-scale constrained optimization that uses minFunc as a sub-routine, available here (I have not used this code).

Updates

September, 2009: Some minor updates involving mex files:

- The 64-bit linux mex file 'lbfgsC.mexa64' was added. Thanks to Emt Khan for this file.
- The 64-bit windows mex files 'lbfgsC.mexw64' and 'mcholC.mexw64' were added, as well as an updated version of 'mcholC.c' that allows this file to be compiled under 64-Windows machines. Thanks to Rainer Heintzmann for these files.
- The Intel Mac mex files 'lbfgsC.mexmaci' was added. Thanks to Jascha Sohl-Dickstein for this file. On January 14, 'lbfgsC.mexmaci64' and 'mCholC.mexmaci64' were added thanks to Jascha.

July, 2009: I put up another update of minFunc (this version is available here). Another set of minor changes in response to feedback, but also some non-trivial changes to the conjugate gradient and dense quasi-Newton methods. Some highlights include:

- Removed bounding of the initial step length in the line search (except on the first iteration). This sometimes caused slow convergence for functions with large gradient values.
- Changed the optimality tolerance parameters; tolFun is now only used to measure first-order optimality conditions, while tolX is now used for measures of lack of progress.
- Added the 'cyclic' steepest descent method, which performs an accurate line search, but then performs several iterations with the same step size under an inaccurate line search.
- To prevent the non-linear conjugate gradient method from restarting so often, this method was modified to accept the conjugate gradient step whenever a sufficient decrease condition is satisfied.
- Added the 'scaled' conjugate gradient method, where a Hessian-vector product is used to give a good initialization of the line search.
- Added the option to use the eigenvector associated with the most negative eigenvalue as a negative curvature direction, as an alternative to Hessian modification strategies.
- When the linear conjugate gradient method terminates because of negative curvature in the newton0 method, the option is now available to use the negative curvature direction as the search direction (now the default).
- Added a preconditioned non-linear conjugate gradient method.
- Added the ability to call user-specified preconditioners.
- Changed the name of the 'bfgs' method to 'qnewton', as there are now a bunch of Hessian approximations available via the 'qnUpdate' option: BFGS updates, SR1 updates (falls back on BFGS if it doesn't maintain positive-definiteness), Hoshino updates, Self-Scaling BFGS (now the default), Oren's Self Scaling Variable Metric Method, and the McCormick-Huang asymmetric update.
- Removed the constrained optimization codes. For constrained optimization, see minConf.

April, 2008: I put an updated version of minFunc on this webpage (this version is available here). Most of the changes made were minor and were made in response to user feedback. Here are some highlights:

- Input options are now case insensitive.
- The Wolfe line search now switches to an Armijo line search if the function returns an illegal (complex, NaN, or Inf) value (previously, this was one of the biggest causes of problems and you had to make the switch manually for any function that you suspected might overflow or have other problems)
- The Wolfe line search is now more robust (thanks to Glenn Fung, Mirela Andronescu, and Frank Hutter for helping me find some pathological cases).
- Added the ability to have user-specified Hessian-vector product functions in the Hessian-Free Newton method.
- Added a mex version of the LDL^T factorization (often the fastest Hessian modification strategy).
- Made finite-differencing the default numerical differentiation technique (this is less accurate but more broadly applicable than the complex-step derivative approach).
- Added a Tensor method that uses a Tensor of 3rd derivatives to potentially give improved convergence (this is mainly for entertainment purposes since I can't imagine it being practically useful).
- Added three constrained optimization codes that use some of minFunc's functionality to solve constrained optimization problems. 'minFuncBC' solves problems with (upper/lower) bounds on the variables, 'minFuncEC' solves problems with linear equality constraints, and 'minFuncIC' solves problems with linear inequality constraints (this one is not as efficient).

June, 2007: Since a bunch of people have been using it, I put a version of minFunc online at this webpage (this first version is still available here).

I have also written Trust-Region and Derivative-Free unconstrained optimization codes. In some scenarios, the

second-order Trust-Region code requires fewer function evaluations than minFunc (based on line-searches), while some of the Derivative-Free methods can be applied to problems that are continuous but not necessarily differentiable (ie. if the function uses absolute values). These methods are not available in the on-line package, contact me if you want these additional methods.

Common Problems

There are several situations that account for the majority of the unsatisfactory behaviour of minFunc (with the first being the most common):

1. The user-supplied gradient code is wrong. It is recommended to use the 'DerivativeCheck' option to make sure that the gradient code roughly matches numerically computed derivatives.
2. The objective function is not appropriate (this is a modeling issue, not an optimization problem). You need to make sure that a minimum of the function is actually the result you want, and that you don't need constraints.
3. minFunc returns before a solution of satisfactory accuracy is achieved. In this case, decreasing TolX and/or TolFun will typically fix the problem, but in some cases the objective function or variables need to be re-scaled.
4. minFunc runs out of memory. By default, minFunc uses a large number of corrections in the L-BFGS method. For problems with a very large number of variables, the 'Corr' parameter should be decreased (ie. if you run out of memory on the 10th iteration, try setting 'Corr' to 9 or lower).
5. When using the pure Newton method, the default options in minFunc assume that the Hessian is positive-definite or close to positive-definite. If this is not the case, the 'hessianModify' option may need to be changed (to 1, 2, or 3, for example) in order to obtain good performance.
6. Each iteration requires many function evaluations. In this case, changing the line search initialization (options.LS_init) from its default value can sometimes improve performance (ie. setting it to 2 will use a quadratic approximation to initialize the line search).
7. You get the error "??? Undefined function or method 'lbfgsC'". This occurs if Matlab can't find the mex file for your operating system. If this occurs, you can either: (i) set 'options.useMex' to 0, or (ii) compile the mex file for your operating system: "mex lbfgsC.c". If you end up compiling a file in minFunc for your operating system, please send it to me and I will add to the distribution.

Other Issues

Sometimes, we want to minimize a function of x that has the interface myFunc(y,x,arg1,arg2) instead of myFunc(x,arg1,arg2), such as when using Matlab's object oriented programming functionality. To switch the argument order, you can use an anonymous function:

```
myObj = @(x)f(y,x,arg1,arg2);
minFunc(myObj,x_init);
```

To reference minFunc in a publication, please include my name and a link to this website. You may also want to include the date, since I may update the software in the future.
