A Short Tutorial to Gurobi

Peng Zeng[*]
Department of Mathematics and Statistics, Auburn University

May 26, 2020

Abstract

Gurobi is a fast mathematical programming solver that can solve linear programming and quadratic programing among others. It provides an easy and simple interface to many programming languages such as R, matlab, and C. This tutorial discusses the installation of Gurobi and its interface to R, matlab, and C.

1 Introduction

Gurobi (`http://www.gurobi.com`) is a fast mathematical programing solver that can solve many optimization problems including linear programming and quadratic programming. It provides a free academic license to almost everyone with an .edu email account.

The standard linear programming problem is as follows.

$$\min_x c^T x \quad \text{subject to } Ax = b, \quad Bx \geq d, \quad \ell_k \leq x_k \leq u_k \tag{1}$$

The standard quadratic programing problem is as follows.

$$\min_x x^T Q x + c^T x \quad \text{subject to } Ax = b, \quad Bx \geq d, \quad \ell_k \leq x_k \leq u_k \tag{2}$$

It is easy to solve linear programming or quadratic programing problems using Gurobi. We only need to specify the matrix Q, the vector c, the linear constraints, and the lower and upper bounds of x. The default setting usually yields satisfactory performance. However, Gurobi does provide options to fine-tune the optimization process if necessary.

This tutorial focuses on the interface of Gurobi to R, matlab, and C.

2 Installation of Gurobi

Gurobi can be downloaded at `http://www.gurobi.com`. After downloading and installing Gurobi, make sure to request a free academic licence, which is valid for one year and renewable.

Remember the home directory and call it `GUROBI_HOME`, which may be needed later. In this directory (or its subdirectory `win64` or `linux64`), there are subdirectories such as matlab and R, which contain packages providing interfaces to those programming languages.

Gurobi uses the following terms across different programming languages.

[*]Peng Zeng, Department of Mathematics and Statistics, Auburn University, Auburn, AL 36830, USA. Email: `zengpen@auburn.edu`.

- `modelsense`: max or min in (1) or (2).
- `obj`: the vector c in (1) or (2).
- `Q`: the matrix Q in (2).
- `A`: the matrix A or B in the linear constraints in (1) or (2).
- `rhs`: the vector b or d in the linear constraints in (1) or (2).
- `sense`: $=$, $\geq$, or $\leq$ in the linear constraints in (1) or (2).

As a demonstration, let us solve the lasso problem using gurobi

$$\min_{\beta} \frac{1}{2}\|y - X\beta\|^2 + \sum_{k=1}^{p} \lambda_k |\beta_k|.$$

where $y \in \mathbb{R}^n$ is the vector of responses, $X \in \mathbb{R}^{n\times p}$ is the design matrix, $\beta \in \mathbb{R}^p$ is the vector of parameters, and $\lambda \in \mathbb{R}^p$ is the vector of tuning parameters. Introducing β^+ and β^- as the positive and negative components of β, the lasso problem becomes

$$\min_{\beta^+,\beta^-} \frac{1}{2}\begin{pmatrix}\beta^+\\ \beta^-\end{pmatrix}^T \begin{pmatrix}X^TX & -X^TX\\ -X^TX & X^TX\end{pmatrix}\begin{pmatrix}\beta^+\\ \beta^-\end{pmatrix} + \begin{pmatrix}\beta^+\\ \beta^-\end{pmatrix}^T \begin{pmatrix}\lambda - X^Ty\\ \lambda + X^Ty\end{pmatrix} + \frac{1}{2}y^Ty$$

subject to $\beta^+ \geq 0$ and $\beta^- \geq 0$.

3 Interface to R

After installing Gurobi successfully, the Gurobi R package file can be found in the subdirectory named R in `GUROBI_HOME`. Read the associated `README` for the instruction of installation. The package `gurobi` can be installed using one of the following ways.

- In command line, run the following command

```
R CMD INSTALL <R-package-file>
```

- In R, run the following command

```
install.packages("<R-package-file>", repos = NULL)
```

- In R, choose `Install package(s) from local files ...` from the Packages menu.

Refer to the associated file `lasso.gurobi.R` for the complete code. We only include the codes related to Gurobi with some comments in the following. Refer to the R documents for more details.

- The function `gurobi()` has two arguments, `model` and `params`. Both are lists.
- `model` is a list containing information on model specifications. It has components such as `modelsense`, `obj`, `lb`, `ub`, `objcon`, `Q`, `A`, `rhs`, `sense`, etc.

- `params` is a list containing information on options that control model fitting algorithms, tuning parameters, output, and more.
- The output `QPfit` is a list with components `status` for the statue of the problem, `x` for the solution to the problem, `objval` for the optimal value of the problem.
- `model$A`, `model$rhs`, `model$sense` cannot be `NULL`. Specify them as matrix/vectors with 0 components if there is no linear constraint.

```
library(gurobi);

model = list(modelsense = 'min');
model$Q = 0.5 * rbind(cbind(xtx, -xtx), cbind(-xtx, xtx));
model$obj = c(lambda - xty, lambda + xty);
model$objcon = 0.5 * sum(y * y);
model$lb = rep(0, p + p);     #  0  is the default, can be removed
model$ub = rep(Inf, p + p);   # inf is the default, can be removed

model$A = matrix(0.0, nrow = 0, ncol = p + p);
model$rhs = rep(0.0, 0);
model$sense = rep('>', 0);

params = list(OutputFlag = 0, ...);
QPfit = gurobi(model, params);
```

4 Interface to Matlab

The interface to matlab is similar to that to R. Just notice that `Q` and `A` must be a sparse matrix.

```
run C:\gurobi811\win64\matlab\gurobi_setup;

bmodel.modelsense = 'min';

bmodel.Q = 0.5 * sparse([Smat, -Smat; -Smat, Smat]);
bmodel.obj = [lambda - avec; lambda + avec];
bmodel.lb = zeros(p + p, 1);
bmodel.ub = inf(p + p, 1);

bmodel.A = sparse(zeros(0, p + p));
bmodel.rhs = zeros(0, 1);
bmodel.sense = repmat('=', 0, 1);

bparams.OutputFlag = 0;

bresult = gurobi(bmodel, bparams);
```

5 Interface to C

The interface to C is complicated. The following are the main steps with some explanation.

- Create a Gurobi environment using `GRBloadenv()`, which checks the Gurobi license and initializes the environment.
- Create a Gurobi model using `GRBnewmodel()`, which specifies the number of variables using `nvar`, the vector c in (1) or (2) using `obj`, and the lower and upper bounds using `lb` and `ub`. Here, `obj`, `lb`, `ub` are all double arrays with length `nvar`.
- Specify the matrix Q using `GRBaddqpterms()`, where `count` is the number of nonzero entries of Q and `Qrow`, `Qcol`, `Qval` specify the locations and values of the nonzero entries of Q. Notice that the Gurobi interface is built around quadratic terms, rather than a Q matrix. If the quadratic objective contains a term $2xy$, it can be entered as a single term, $2xy$, or as a pair of terms, xy and yx.
- Need to call `GRBupdatemodel()` before fitting the model using `GRBoptimize()`.
- Make sure to release the memory using `GRBfreemodel()` and `GRBfreeenv()` in the end.

```
#include <gurobi_c.h>

GRBenv *env = NULL;
GRBloadenv(&env, NULL);

GRBmodel *model = NULL;
GRBnewmodel(env, &model, "lasso_gurobi", nvar, obj, lb, ub, NULL, NULL);

GRBaddqpterms(model, count, Qrow, Qcol, Qval);

GRBsetintparam(GRBgetenv(model), "OutputFlag", 0);
GRBupdatemodel(model);
GRBoptimize(model);

GRBfreemodel(model);
GRBfreeenv(env);
```

It is important to specify the correct location of library and header files. When calling C in R, create a file named `Makevars` with the following lines.

```
GUROBI_LIBS = -L"$(GUROBI_HOME)\lib" -lgurobi81
PKG_CPPFLAGS = -I"$(GUROBI_HOME)\include"
PKG_LIBS = $(LAPACK_LIBS) $(BLAS_LIBS) $(GUROBI_LIBS)
```

6 Conclusion

Gurobi is an easy to use and fast mathematical programming solver. It provide detailed documents on `https://www.gurobi.com/documentation/`. The documents explain almost everything you need to know about Gurobi.