CS 240H Project Report: Linear Least Squares Modeling in Haskell

Tri Dao
trid@stanford.edu

March 18, 2016

1 Introduction

Least squares are optimization problems that minimize the norm square of an affine expression, subject to linear constraints on the variables. They have the general form

$$\begin{array}{ll} \text{minimize} & \|Ax - b\|^2 \\ \text{subject to} & Cx = d \end{array} \tag{1}$$

where $x \in \mathbb{R}^n$ is a variable, $A \in \mathbb{R}^{m \times n}$ and $C \in \mathbb{R}^{p \times n}$ are matrices, and $b \in \mathbb{R}^m$ and $d \in \mathbb{R}^p$ are vectors.

These problems are routinely solved in the context of statistical estimation, machine learning, and engineering design. For example, linear regression, an approach for modeling the relationship between a scalar dependent variable y and one or more explanatory variables X, involves solving the problem minimize $\|X\beta - y\|^2$ to find β, where X is a given data matrix, y is a given vector, and β is the *parameter vector*.

In general, there can be many variables, the objective can contain many quadratic terms, and there can be multiple linear constraints. Typically one has to rewrite the problem in the standard form (1) and then use a general solver implemented by many numerical linear algebra packages. However, this is a tedious and error-prone process, especially for problems with many variables or several linear constraints.

We aim to develop a modeling language embedded in Haskell to solve these linearly constraints least squares problems. This modeling language (in Haskell) will allow user to specify the problem in a natural way that mirrors standard mathematical notation without being constrained by the standard form.

2 Background

The (unconstrained) *least squares problem* has the standard form

$$\text{minimize} \quad \|Ax - b\|^2,$$

where $x \in \mathbb{R}^n$ is a variable, $A \in \mathbb{R}^{m \times n}$ is a matrix, and $b \in \mathbb{R}^m$ is a vector. Assuming that A is full rank, the least squares problem has the close-form solution

$$\hat{x} = (A^T A)^{-1} A^T b.$$

A slight generalization is the (linearly) *constrained least squares problem*:

$$\begin{array}{ll} \text{minimize} & \|Ax - b\|^2 \\ \text{subject to} & Cx = d \end{array}$$

where $x \in \mathbb{R}^n$ is a variable, $A \in \mathbb{R}^{m \times n}$ and $C \in \mathbb{R}^{p \times n}$ are matrices, and $b \in \mathbb{R}^m$ and $d \in \mathbb{R}^p$ are vectors. The solution of this constrained problem is

$$\begin{bmatrix} \hat{x} \\ z \end{bmatrix} = \begin{bmatrix} 2A^T A & C^T \\ C & 0 \end{bmatrix}^{-1} \begin{bmatrix} 2A^T b \\ d \end{bmatrix},$$

where z is the dual variable to the equality constraint. We assume that the matrix above is invertible. This occurs when the matrix C has independent rows, and the matrix $\begin{bmatrix} A \\ C \end{bmatrix}$ has independent columns.

3 Modeling

The key data structure is an `Expr`, which contains variables and constants that are combined using linear mathematical functions (addition, scaling, matrix-vector multiplication, etc.).

4 Implementation

Given a problem whose objective is an Expression and whose constraints are multiple other Expressions, the system will analyze the abstract syntax tree to transform it to the standard form above, and then solve it using linear algebra subroutines.

The three steps of solving a least squares problem is given in Figure 1.

5 Examples

6 Conclusion

We have designed and implemented a modeling language embedded in Haskell for linearly constrained least squares problem. There are still some work to be done to add more linear operators (vector indexing, convolution, Fourier transform, vector and matrix stacking, element wise multiplication, etc.). Moreover, the package only supports dense matrices due to lack of sparse matrix support in Haskell.

Overall, this modeling language will allow users to express their problems in a natural form while the system takes care of transforming it to the standard form. This might lead to faster prototyping and data analysis in statistical estimation, machine learning, and engineering design.

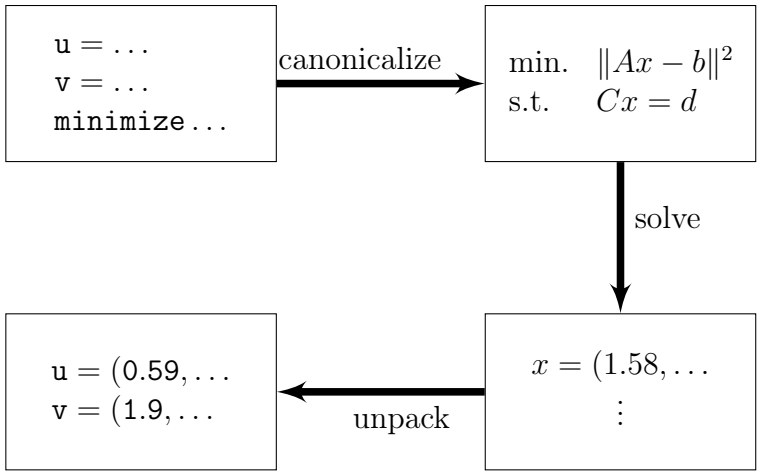

Figure 1: Three steps of solving a least squares problem.

Acknowledgment

We thank Professor Stephen Boyd for introducing us to many examples of least squares modeling in control, statistical estimation, and engineering design. We also thank David Zeng whose Julia package `LinearLeastSquares` inspired this project.