Seminar: Techniques for implementing main memory databases

Linear and logistic regression with SQL

Oleg Patrascu

23.12.2017

Abstract

Databases are huge containers of data and are great at aggregating it but when it comes to statistical analysis they lack a lot of features. Most of the statistical functionality is handled by an additional layer of software like R or Matlab. We will see if linear or logistic regression is possible with SQL and provide their implementation and enough arguments whether an SQL implementation of these is worth, or maybe it's not worth to perform any numerical calculations on the database side.

1 Simple Linear Regression

Simple linear regression is a statistical method that allows us to summarize and study relationships between two continuous (quantitative) variables: One variable, denoted x, is regarded as the predictor, explanatory, or independent variable. The other variable, denoted y, is regarded as the response, outcome, or dependent variable.

Given the training set $\{y_i, x_i\}_{i=1}^n$ we try to find the best coefficients of the equation $y_i = \beta_0 + \beta_1 x_i$ so that we approximate $\hat{y}_i \approx y_i$ according to ordinary least squares (OLS) : conceptually the simplest and computationally straightforward which is advantageous for SQL.

So if we take the derivative of $\sum_{i=1}^n (y_i - \hat{y}_i)^2$ with respect to β_0 and β_1 we get

$$\beta_0 = (\bar{y} - \beta_1 \bar{x})$$

$$\beta_1 = \frac{\sum_{i=1}^{n}(y_i - \bar{y})(x_i - \bar{x})}{\sum_{i=1}^{n}(x_i - \bar{x})^2}$$

- With only 2 parameters to estimate : β_0, β_1 computationally it is not a challenge for any DBMS
- But increasing the parameters it will slow by noticeable constant factor if we solve all β in a "scalar" fashion .
- However we could use a some matrix tricks to solve for any size of β

Implementing the equations above in SQL is fairly easy.

```
table
-----
X (numeric)
Y (numeric)

/**
 * m = (nSxy - SxSy) / (nSxx - SxSx)
 * b = Ay - (Ax * m)
 * N.B. S = Sum, A = Mean
 */
DECLARE @n INT
SELECT  @n = COUNT(*) FROM table
SELECT (@n * SUM(X*Y) - SUM(X) * SUM(Y)) /
                        (@n * SUM(X*X) - SUM(X) * SUM(X)) AS M,
       AVG(Y) - AVG(X) *
       (@n * SUM(X*Y) - SUM(X) * SUM(Y)) /
      (@n * SUM(X*X) - SUM(X) * SUM(X)) AS B
FROM table
```

2 Multiple linear regression

Simple Linear regression alone is not enough because estimating only one parameter is very easy and doesn't bring much benefit as opposed to estimating a vector of parameters. So scaling up the linear regression problem we bring its use to today's problems.

$$y = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix} \quad X = \begin{bmatrix} 1 & x_{11} & x_{12} & x_{13} & \dots & x_{1p} \\ 1 & x_{21} & x_{22} & x_{23} & \dots & x_{2p} \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_{n1} & x_{n2} & x_{n3} & \dots & x_{np} \end{bmatrix} \quad \beta = \begin{bmatrix} \beta_0 \\ \beta_1 \\ \vdots \\ \beta_p \end{bmatrix}$$

$$\hat{y} = X\beta$$

$$LS(\beta) = (y - X\beta)^T(y - X\beta)$$

Minimizing the above function (taking the first derivative according to β) will yield :

$$\hat{y} = X\beta = X(X^TX)^{-1}X^Ty$$

QR Factorization simplifies the process but matrix multiplication in pure SQL is a pain and also inneficient. The QR factorization is used by R's $lm()$ function (implemented with C and Fortran calls underneath).

Operations for linear algebra must be implemented :

- Matrix/vector multiplication
- Matrix inverse/pseudo-inverse
- Matrix factorization like SVD, QR, Cholesky factorization

Too much number crunching for an engine which has different purpose, even C++'s Boost library for linear algebra (BLAST) does a poor job in comparison to Matlab.

3 Logistic Regression

Predicts an outcome variable that is categorical from predictor variables that are continuous and/or categorical. Used because having a categorical outcome variable violates the assumption of linearity in normal regression. It expresses the linear regression equation in logarithmic terms (called the logit).

In order to keep the outcome between 0 and 1, we apply the logistic function (also called sigmoid function):

$$g(z) = \frac{1}{1 + exp(-z)}$$

Bringing this to more general vector input :

$$h_\beta(x) = \frac{1}{1 + e^{-\beta^T x}} = P(y = 1|x; \beta)$$

So after this function will take as input our data it will yield a "yes" or "no" answer. And again we dive into linear algebra operations making the SQL implementation almost impossible.

4 Implementation

Simple linear regression was done efortlessly. The multiple linear regression and logistic regression were implemented using armadillo library. Armadillo is a high quality linear algebra library (matrix maths) for the C++ language, aiming towards a good balance between speed and ease of use, it provides high-level syntax (API) deliberately similar to Matlab.

$$MultipleLinearRegression$$

```
arma::colvec fastLM(const arma::vec & y, const arma::mat & X) {
int n = X.n_rows, k = X.n_cols;

arma::colvec coef = arma::solve(X, y);    // fit model y ~ X
arma::colvec res  = y - X*coef;           // residuals

// std.errors of coefficients
double s2 =
std::inner_product(res.begin(), res.end(), res.begin(), 0.0)/(n - k);

arma::colvec std_err =
arma::sqrt(s2 * arma::diagvec(arma::pinv(arma::trans(X)*X)));

return List::create(Named("coefficients") = coef,
       Named("stderr") = std_err,
       Named("df.residual")  = n - k);
```

5 Conclusion

The only kind of regression which is worth to implement directly in SQL would be the simple linear regression due to less overhead. Multiple and logistic require complex operations of a numerical nature and this is very demanding from a non turing complete language like SQL. The database engine could of course offer such statistical functionality out of the box, implemented underneath in C/C++ but for databases like Postgres, MySQL, DB2 it would be a pollution mainly because that is not their purpose.

It would be reasonable as well to ask the question : Where should such kind of calculations be performed, on the database side or server side ? As long as there is no complex linear algebra operations involved, than the database side is the winner, not only because it has an optimizer to aggregate data very fast but also the data must not passed between server and database. The types of databases which could deal easily with requirements like Sqream or Kinetica which are tailored for big data queries.

References

[1] Wikipedia: Logistic Regression, `https://en.wikipedia.org/wiki/Linear_regression`

[2] Wikipedia: Logistic Regression, `https://en.wikipedia.org/wiki/Logistic_regression`

[3] SO: Dabatase Calculations vs Server Calculations, `https://stackoverflow.com/questions/6449072/doing-calculations-in-mysql-vs-php`