Linear Regression II

Multivariate Scalar Models

Uwe Naumann

Informatik 12:
Software and Tools for Computational Engineering (STCE)

RWTH Aachen University

Contents

Outline

Objective

- Introduction to linear regression methods for multivariate scalar models.

Learning Outcomes

- You will understand
  - normal equation
  - Givens rotation
  - Householder reflection.
- You will be able to
  - implement linear regression methods
  - run computational experiments.

Outline

A linear (in $\mathbf{p}$) multivariate scalar model

$$y = f(\mathbf{p}, \mathbf{x}) = \mathbf{g}(\mathbf{x})^T \cdot \mathbf{p} \,, \quad \text{for } \mathbf{g} : \boldsymbol{R}^n \rightarrow \boldsymbol{R}^n$$

yields the linear regression problem

$$A \cdot \mathbf{p} \approx \mathbf{y}$$

for given data $X \in \boldsymbol{R}^{m \times n}, \mathbf{y} \in \boldsymbol{R}^m, m \geq n$ and $A = \left(\mathbf{g}(X_i^T)^T\right) \in \boldsymbol{R}^{m \times n}$.
Minimization of the error

$$E(\mathbf{p}) = \|F(\mathbf{p}, X, \mathbf{y})\|_2^2 = \sum_{i=0}^{m-1} F_i(\mathbf{p}, X, \mathbf{y})^2 = \sum_{i=0}^{m-1} (f(\mathbf{p}, A_i(X_i^T)) - y_i)^2$$

$$= \sum_{i=0}^{m-1} (\mathbf{g}(X_i^T)^T \cdot \mathbf{p} - y_i)^2 = \|A \cdot \mathbf{p} - \mathbf{y}\|_2^2$$

can be regarded as a convex minimization problem (see module Newton_II).
Exploitation of special problem structure yields potentially more efficient solution methods.

We consider three approaches to the solution of the linear regression problem

$$A \cdot \mathbf{p} \approx \mathbf{y} \;, \quad \text{where } A = (a_{i,j})_{j=0,\ldots,n-1}^{i=0,\ldots,m-1} \in \boldsymbol{R}^{m \times n},\ \mathbf{p} \in \boldsymbol{R}^{n},\ \mathbf{y} \in \boldsymbol{R}^{m} :$$

- normal equation
- Givens rotation
- Householder reflection

The normal equation method involves squaring of (uncertain, erroneous) values which may have a negative impact on numerical stability as errors also get squared.

Householder projection and the normal equations method exhibit about the same computational cost. The former is typically more accurate.

So is the Givens rotation method which allows for more selective generation of zeros, and hence may result in superior computational cost for sparse problems.

Outline

The scalar linear regression problem $\mathbf{a} \cdot p \approx \mathbf{y}$ was introduced in module LinearRegression_I. The first order optimality condition

$$\frac{dE}{dp} = \frac{d\|\mathbf{a} \cdot p - \mathbf{y}\|_2^2}{dp} = 0$$

yielded the linear normal equation $\mathbf{a}^T \cdot \mathbf{a} \cdot p = \mathbf{a}^T \cdot \mathbf{y}$ with trivial solution

$$p = \frac{\mathbf{a}^T \cdot \mathbf{y}}{\mathbf{a}^T \cdot \mathbf{a}} .$$

Satisfaction of the second-order optimality condition

$$\frac{d^2 E}{dp^2} = 2 \cdot \mathbf{a}^T \cdot \mathbf{a} > 0$$

for $\mathbf{a} \neq \mathbf{0} \in R^m$ implied a local minimum at p.

The optimality conditions for minimization of the error function

$$E = \|A \cdot \mathbf{p} - \mathbf{y}\|_2^2 = \sum_{i=0}^{m-1} (\mathbf{a}_i \cdot \mathbf{p} - y_i)^2 \quad (\in \boldsymbol{R}) \,,$$

where $\mathbf{a}_i \in \boldsymbol{R}^{1\times n}$ denotes the i-th row of A, require the gradient

$$E' = \sum_{i=0}^{m-1} (2 \cdot (\mathbf{a}_i \cdot \mathbf{p} - y_i) \cdot \mathbf{a}_i) \quad (\in \boldsymbol{R}^{1\times n})$$

to vanish identically with positive definite Hessian

$$E'' = 2 \cdot \sum_{i=0}^{m-1} \mathbf{a}_i^T \cdot \mathbf{a}_i = 2 \cdot A^T \cdot A \,.$$

The latter is satisfied due to strict convexity of E.

From

$$E'^{T} = 2 \cdot \sum_{i=0}^{m-1} \mathbf{a}_i^T \cdot (\mathbf{a}_i \cdot \mathbf{p} - y_i) = 0$$

follows

$$\sum_{i=0}^{m-1} \mathbf{a}_i^T \cdot \mathbf{a}_i \cdot \mathbf{p} - \mathbf{a}_i^T \cdot y_i = 0$$

yielding the normal equation

$$A^T \cdot A \cdot \mathbf{p} = A^T \cdot \mathbf{y}$$

which can be solved by LL^T (LDL^T) factorization.

Note: Matrix product $A^T \cdot A$ as sum of outer products of indvidual rows of A.

```
template<typename T, int M, int N>
void NormalEquation(
    const Eigen::Matrix<T,M,N> &A,
    Eigen::Matrix<T,N,1> &p,
    const Eigen::Matrix<T,M,1> &y
) {
  p=(A.transpose()*A).llt().solve(A.transpose()*y);
}

int main(int argc, char* argv[]) {
  assert(argc==3); int m=std::stoi(argv[1]), n=std::stoi(argv[2]);
  using T=double;
  using MT=Eigen::Matrix<T,Eigen::Dynamic,Eigen::Dynamic>;
  using VT=Eigen::Matrix<T,Eigen::Dynamic,1>;
  MT A=MT::Random(m,n); VT p(n), y=VT::Random(m);
  NormalEquation(A,p,y);
  std::cout << "p=" << p.transpose()<< std::endl;
  return 0;
}
```

Outline

Orthogonalization transforms the scalar linear regression problem $\mathbf{a} \cdot p \approx \mathbf{y}$ into

$$Q \cdot \begin{pmatrix} \|\mathbf{a}\|_2 \\ 0 \\ \vdots \\ 0 \end{pmatrix} \cdot p \approx \mathbf{y}$$

using an orthogonal matrix $Q \in R^{m \times m}$ $(Q^T = Q^{-1})$. The resulting linear equation

$$\|\mathbf{a}\|_2 \cdot p = \left[Q^T \cdot \mathbf{y}\right]_0 \ ,$$

where $[\mathbf{v}]_0$ denotes the first entry of a vector $\mathbf{v}$, has the unique solution

$$p := \frac{\left[Q^T \cdot \mathbf{y}\right]_0}{\|\mathbf{a}\|_2} \ .$$

Givens rotation or Householder reflection can be used to construct Q.

Orthogonalization of the linear regression problem $A \cdot \mathbf{p} \approx \mathbf{y}$ yields

$$Q \cdot R \cdot \mathbf{p} \approx \mathbf{y}$$

with orthogonal $Q \in \boldsymbol{R}^{m \times m}$ and upper triangular $R \in \boldsymbol{R}^{m \times n}$.

Note that column-wise orthogonalization for $j = 0, \ldots, n-1$ ensures that columns $k < j$ are not affected.

The resulting system of linear equations

$$[R]_{0,\ldots,n-1} \cdot \mathbf{p} = \left[Q^T \cdot \mathbf{y}\right]_{0,\ldots,n-1} ,$$

where $[M]_{0,\ldots,n-1}$ denotes the first n rows of a matrix M, can be solved by backward substitution.

Givens rotation or Householder reflection construct $Q^{-1} = Q^T \in \mathbf{R}^{n \times n}$ as a matrix chain product with factors rotating / reflecting the j-th column

$$\begin{pmatrix} a_{0,j} \\ \vdots \\ a_{j,j} \\ \vdots \\ a_{m-1,j} \end{pmatrix} \text{ of } A \text{ into } \begin{pmatrix} a_{0,j} \\ \vdots \\ \sqrt{\sum_{i=j}^{m-1} a_{i,j}^2} \\ 0 \\ \vdots \\ 0 \end{pmatrix}$$

for $j = 0, \ldots, n-1$. The matrix $A \in \mathbf{R}^{m \times n}$ is transformed into the upper triangular matrix $R \in \mathbf{R}^{m \times n}$ by a sequence of orthogonal rotation / projections $Q_i^T \in \mathbf{R}^{n \times n}$.

Software and Tools for Computational Engineering · RWTH AACHEN UNIVERSITY

$$G_{i,j}^k \equiv \begin{pmatrix} 1 & & & & & & \\ & \ddots & & & & & \\ & & \frac{a_{i,k}}{\sqrt{a_{i,k}^2+a_{j,k}^2}} & \cdots & \frac{a_{j,k}}{\sqrt{a_{i,k}^2+a_{j,k}^2}} & & \\ & & & \ddots & & & \\ & & \vdots & 1 & \vdots & & \\ & & & & \ddots & & \\ & & -\frac{a_{j,k}}{\sqrt{a_{i,k}^2+a_{j,k}^2}} & \cdots & \frac{a_{i,k}}{\sqrt{a_{i,k}^2+a_{j,k}^2}} & & \\ & & & & & \ddots & \\ & & & & & & 1 \end{pmatrix}$$

transforms the k-th column $\mathbf{a}$ of $A \in R^{m \times n}$ into $\mathbf{b} = G_{i,j}^k \cdot \mathbf{a}$ such that $\|\mathbf{b}\|_2 = \|\mathbf{a}\|_2$ and $b_j = 0$. It performs an orthogonalization of $\mathbf{a}$ wrt. the Cartesian basis direction $\mathbf{e}_j$ as $\mathbf{b}^T \cdot \mathbf{e}_j = 0$.

Givens rotation $G^j_{i,i+1}$ applied to all columns $j = 0, \ldots, n-1$ of $A \in R^{m \times n}$ yield transformations

$$G^j_{i,i+1} \cdot \begin{pmatrix} \vdots \\ a_{i,j} \\ a_{i+1,j} \\ \vdots \end{pmatrix} \Rightarrow \begin{pmatrix} \vdots \\ \sqrt{a^2_{i,j} + a^2_{i+1,j}} \\ 0 \\ \vdots \end{pmatrix}$$

for $i = m-2, \ldots, j$ implying

$$G^{n-1}_{n,n+1} \cdot \ldots \cdot G^{n-1}_{m-2,m-1} \cdot \ldots \cdot G^0_{0,1} \cdot \ldots \cdot G^0_{m-2,m-1} \cdot A = R$$

and, hence, from $Q \cdot R = A$

$$Q^{-1} = Q^T = G^{n-1}_{n,n+1} \cdot \ldots \cdot G^{n-1}_{m-2,m-1} \cdot \ldots \cdot G^0_{0,1} \cdot \ldots \cdot G^0_{m-2,m-1} \cdot$$

Note that columns $k < j$ are not affected as zeros are rotated.

Givens Rotation

Implementation

```
template<typename T, int M, int N>
void Givens(const Eigen::Matrix<T,M,N> &A,
    Eigen::Matrix<T,M,M> &QT, Eigen::Matrix<T,M,N> &R) {
  int m=A.rows(), n=A.cols();
  using MT=Eigen::Matrix<T,M,M>;
  R=A; QT=MT::Identity(m,m);
  for (int j=0;j<n;j++) {
    for (int i=m-2;i>=j;i--) {
      T norm_a_tilde=R.col(j).block(i,0,2,1).norm();
      MT G=MT::Identity(m,m);
      G(i,i)=R.col(j)(i)/norm_a_tilde; G(i+1,i+1)=G(i,i);
      G(i,i+1)=R.col(j)(i+1)/norm_a_tilde; G(i+1,i)=-G(i,i+1);
      QT=G*QT; R=G*R;
    }
  }
}
```

Implementation

```
template<typename T, int M, int N>
void LinearSolve(const Eigen::Matrix<T,M,M> &QT, const Eigen::Matrix<T,M,N> &R,
    Eigen::Matrix<T,N,1> &p, const Eigen::Matrix<T,M,1> &y) {
  int n=R.cols();
  Eigen::Matrix<T,M,1> r=QT*y;
  for (int i=n-1;i>=0;i--) {
    T d=r(i);
    for (int j=n-1;j>i;j--) d-=R(i,j)*p(j);
    p(i)=d/R(i,i);
  }
}

template<typename T, int M, int N>
void Regression(const Eigen::Matrix<T,M,N> &A,
    Eigen::Matrix<T,N,1> &p, const Eigen::Matrix<T,M,1> &y) {
  int m=A.rows(), n=A.cols();
  Eigen::Matrix<T,M,M> QT(m,m); Eigen::Matrix<T,M,N> R(m,n);
  Givens(A,QT,R);
  LinearSolve(QT,R,p,y);
}
```

Householder Reflection

Recall

Householder reflection of a vector

$$\mathbf{a} = \begin{pmatrix} a_0 \\ \vdots \\ a_{m-1} \end{pmatrix} \in \boldsymbol{R}^m$$

yields

$$H \cdot \mathbf{a} = \left(I - 2 \cdot \frac{\mathbf{v} \cdot \mathbf{v}^T}{\mathbf{v}^T \cdot \mathbf{v}} \right) \cdot \mathbf{a} = \|\mathbf{a}\|_2 \cdot \mathbf{e}_0 = \begin{pmatrix} \sqrt{\sum_{i=0}^{m-1} a_i^2} \\ 0 \\ \vdots \\ 0 \end{pmatrix}$$

for $\mathbf{v} = \mathbf{a} - \|\mathbf{a}\|_2 \cdot \mathbf{e}_0$.

Householder Reflection

Nonscalar Case

Products of the symmetric orthogonal Householder matrices H^j with the n columns of A ($j = 0, \ldots, n-1$) yield transformations

$$H^j \cdot \begin{pmatrix} \vdots \\ a_{j,j} \\ a_{j+1,j} \\ \vdots \\ a_{m-1,j} \end{pmatrix} \Rightarrow \begin{pmatrix} \vdots \\ \sqrt{\sum_{i=j}^{m-1} a_{i,j}^2} \\ 0 \\ \vdots \\ 0 \end{pmatrix}$$

(entries of strictly upper triangular submatrix of A remain unchanged) implying

$$H^{n-1} \cdot \ldots \cdot H^0 \cdot A = R$$

and, hence, from $Q \cdot R = A$, $Q^{-1} = Q^T = Q = H^{n-1} \cdot \ldots \cdot H^0$.

Note that columns $k < j$ are not affected as zero vectors are projected.

In order for H^j to act exclusively on the trailing $m-j$ elements of columns in A its leading principal submatrix must be equal to the identity in $\boldsymbol{R}^{j\times j}$, that is,

$$H^j = \begin{pmatrix} 1 & & & \\ & \ddots & & \\ & & 1 & \\ & & & \tilde{H}^j \end{pmatrix},$$

where $\tilde{H}^j \in \boldsymbol{R}^{(m-j)\times(m-j)}$ denotes the Householder matrix for the subdiagonal part of the j-th column of A.

Householder Reflection

Implementation

```
template<typename T, int M, int N>
void Householder(const Eigen::Matrix<T,M,N> &A,
    Eigen::Matrix<T,M,M> &QT, Eigen::Matrix<T,M,N> &R) {
  int m=A.rows(), n=A.cols();
  using VT=Eigen::Matrix<T,M,1>;
  using MT=Eigen::Matrix<T,M,M>;
  R=A; QT=MT::Identity(m,m);
  for (int j=0;j<n;j++) {
    VT x=R.col(j);
    for (int i=0;i<j;i++) x(i)=0;
    VT v=x+x(j)/fabs(x(j))*x.norm()*VT::Unit(m,j);
    MT H=MT::Identity(m,m)-2*v*v.transpose()/v.dot(v);
    QT=H*QT; R=H*R;
  }
}
```

Householder Reflection

Implementation

```
template<typename T, int M, int N>
void LinearSolve(const Eigen::Matrix<T,M,M> &QT, const Eigen::Matrix<T,M,N> &R,
    Eigen::Matrix<T,N,1> &p, const Eigen::Matrix<T,M,1> &y) {
  int n=R.cols();
  Eigen::Matrix<T,M,1> r=QT*y;
  for (int i=n-1;i>=0;i--) {
    T d=r(i);
    for (int j=n-1;j>i;j--) d-=R(i,j)*p(j);
    p(i)=d/R(i,i);
  }
}

template<typename T, int M, int N>
void Regression(const Eigen::Matrix<T,M,N> &A,
    Eigen::Matrix<T,N,1> &p, const Eigen::Matrix<T,M,1> &y) {
  int m=A.rows(), n=A.cols();
  Eigen::Matrix<T,M,M> QT(m,m); Eigen::Matrix<T,M,N> R(m,n);
  Householder(A,QT,R);
  LinearSolve(QT,R,p,y);
}
```

Outline

Summary

- Linear regression methods for multivariate scalar models based on
  - normal equation
  - Givens rotation
  - Householder reflection

Next Steps

- Play with sample code.
- Compare results with convex minimization methods.
- Continue the course to find out more ...