Software and Tools for Computational Engineering

RWTH AACHEN UNIVERSITY

Linear Regression I

Univariate Scalar Models

Uwe Naumann

Informatik 12:
Software and Tools for Computational Engineering

RWTH Aachen University

Contents

Outline

Objective and Learning Outcomes

Objective

- Introduction to linear regression methods for univariate 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 p) univariate scalar model

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

yields the linear regression problem

$$\mathbf{g}(\mathbf{x}) \cdot p \approx \mathbf{y} \text{ , for given data } \mathbf{x}, \mathbf{y} \in \boldsymbol{R}^m \text{ and } \mathbf{g} = (g) \in \boldsymbol{R}^m \,.$$

Minimization of the error

$$E(p) = \|F(p, \mathbf{x}, \mathbf{y})\|_2^2 = \sum_{i=0}^{m-1} F_i(p, \mathbf{x}, \mathbf{y})^2$$

$$= \sum_{i=0}^{m-1} (f(p, x_i) - y_i)^2 = \sum_{i=0}^{m-1} (g(x_i) \cdot p - y_i)^2 = \|\mathbf{a} \cdot p - \mathbf{y}\|_2^2$$

where $\mathbf{a} = (g(x_0) \ \ldots \ g(x_{m-1}))^T$, can be regarded as a convex minimization problem (see module Newton_I). Exploitation of special problem structure yields potentially more efficient solution methods.

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

$$\mathbf{a} \cdot p \approx \mathbf{y} \quad :$$

- 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 first order optimality condition

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

yields the linear normal equation

$$\langle \mathbf{a}, \mathbf{a} \rangle \cdot p = \langle \mathbf{a}, \mathbf{y} \rangle$$

which can be solved trivially in the 1D case:

$$p = \frac{\langle \mathbf{a}, \mathbf{y} \rangle}{\langle \mathbf{a}, \mathbf{a} \rangle} = \frac{\sum_{i=0}^{m-1} g(x_i) \cdot y_i}{\sum_{i=0}^{m-1} g(x_i)^2} \ .$$

$$
\begin{aligned}
0 = \frac{dE(p)}{dp} = \frac{d\|\mathbf{a}\cdot p - \mathbf{y}\|_2^2}{dp} &= \frac{d\left[\sum_{i=0}^{m-1}(a_i \cdot p - y_i)^2\right]}{dp} \\
&= 2 \cdot \sum_{i=0}^{m-1} a_i \cdot (a_i \cdot p - y_i) = \sum_{i=0}^{m-1} a_i^2 \cdot p - a_i \cdot y_i \\
&= p \cdot \sum_{i=0}^{m-1} a_i^2 - \sum_{i=0}^{m-1} a_i \cdot y_i = p \cdot \langle \mathbf{a}, \mathbf{a} \rangle - \langle \mathbf{a}, \mathbf{y} \rangle
\end{aligned}
$$

implies

$$
p = \frac{\langle \mathbf{a}, \mathbf{y} \rangle}{\langle \mathbf{a}, \mathbf{a} \rangle} \quad \text{or} \quad p = \frac{\langle \mathbf{x}, \mathbf{y} \rangle}{\langle \mathbf{x}, \mathbf{x} \rangle} \text{ if } g(x_i) = x_i \ .
$$

Normal Equation

Implementation

```
template<typename T, int M>
Eigen::Matrix<T,M,1> g(const Eigen::Matrix<T,M,1> &x) { return x; }

template<typename T, int M>
T NormalEquation(const Eigen::Matrix<T,M,1> &x, Eigen::Matrix<T,M,1> &y) {
  return y.dot(x)/x.dot(x);
}

int main(int argc, char* argv[]) {
  assert(argc==2); auto M=std::stoi(argv[1]);
  using T=float;
  using VT=Eigen::Matrix<T,Eigen::Dynamic,1>;
  VT x=VT::Random(M), y=VT::Random(M);
  x=x.cwiseProduct(x); // make x_i positive
  T p=NormalEquation(g(x),y);
  std::cout << "p=" << p << std::endl;
  return 0;
}
```

Outline

Both rotation and reflection are used as transformations of

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

into

$$p \cdot \|\mathbf{a}\|_2 \cdot \mathbf{e}_0 = p \cdot M \cdot \mathbf{a} \approx M \cdot \mathbf{y}$$

(rotation or reflection of **a** onto the x-axis implies orthogonality to remaining Cartesian basis directions) using orthogonal matrices $M \in \boldsymbol{R}^m \times \boldsymbol{R}^m$.

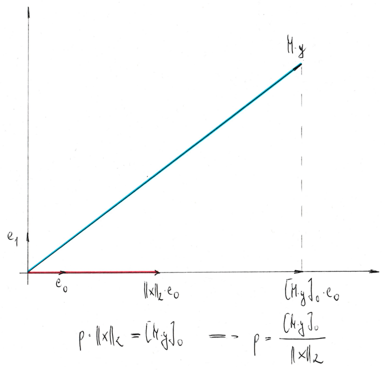
Orthogonalization methods are motivated by the following result:

$$p \cdot \|\mathbf{a}\|_2 \cdot \mathbf{e}_0 \approx M \cdot \mathbf{y} \quad \Rightarrow \quad p = \frac{[M \cdot \mathbf{y}]_0}{\|\mathbf{a}\|_2}$$

where $[M \cdot \mathbf{y}]_i$ denotes the i-th entry of the vector defined by the bracketed expression. For simplicity we write $[\mathbf{v}]_i \equiv v_i$.

The minimal distance between $M \cdot \mathbf{y}$ and a Cartesian basis direction is equal to the geometric projection of $M \cdot \mathbf{y}$ onto that direction:

$$p \cdot \|\mathbf{a}\|_2 \cdot \mathbf{e}_0 \approx M \cdot \mathbf{y} \quad \Rightarrow \quad p = \frac{[M \cdot \mathbf{y}]_0}{\|\mathbf{a}\|_2}$$



The linear operator (matrix) $L \in \mathbf{R}^{m \times m} : \mathbf{R}^m \rightarrow \mathbf{R}^m$ used, e.g, in LR factorization and defined as

$$L \equiv \begin{pmatrix} 1 & & & \\ -\frac{a_1}{a_0} & 1 & & \\ \vdots & & \ddots & \\ -\frac{a_{m-1}}{a_0} & & & 1 \end{pmatrix}$$

transforms a given vector $\mathbf{a} \in \mathbf{R}^m$ into $L \cdot \mathbf{a} = a_0 \cdot \mathbf{e}_0$, e.g,

$$\mathbf{y} = L \cdot \mathbf{x} = \begin{pmatrix} 1 & 0 \\ -\frac{3}{4} & 1 \end{pmatrix} \cdot \begin{pmatrix} 4 \\ 3 \end{pmatrix} = \begin{pmatrix} 4 \\ 0 \end{pmatrix}$$

L is atomic, i.e, $L^{-1} = 2 \cdot I_m - L$ as

$$L \cdot L^{-1} = L \cdot (2 \cdot I_m - L) = 2 \cdot L - L \cdot L = 2 \cdot L - (2 \cdot L - I_m) = I_m \ .$$

L is not orthogonal.

Outline

Orthogonalization

Givens Rotation

The linear operator (matrix) $G \in \boldsymbol{R}^{m \times m} : \boldsymbol{R}^m \rightarrow \boldsymbol{R}^m$ defined as

$$G \equiv \begin{pmatrix} 1 & & & & & & & & \\ & \ddots & & & & & & & \\ & & \frac{a_i}{\|\tilde{\mathbf{a}}\|} & & \cdots & & \frac{a_j}{\|\tilde{\mathbf{a}}\|} & & \\ & & & \ddots & & & & & \\ & & \vdots & & 1 & & \vdots & & \\ & & & & & \ddots & & & \\ & & -\frac{a_j}{\|\tilde{\mathbf{a}}\|} & & \cdots & & \frac{a_i}{\|\tilde{\mathbf{a}}\|} & & \\ & & & & & & & \ddots & \\ & & & & & & & & 1 \end{pmatrix},$$

where $\tilde{\mathbf{a}} \equiv (a_i \; a_j)^T$, transforms $\mathbf{a} \in \boldsymbol{R}^n$ into $\mathbf{b} = G \cdot \mathbf{a}$ such that $\|\mathbf{b}\| = \|\mathbf{a}\|$ and $b_j = 0$. It performs an orthogonalization of $\mathbf{a}$ wrt. the Cartesian basis direction $\mathbf{e}_j$ as $\mathbf{b}^T \cdot \mathbf{e}_j = 0$.

G is orthogonal, i.e, $G^{-1} = G^T$.

Apart from inner Cartesian vector products (yielding unit diagonal entries and vanishing off-diagonal entries) two inner vector products yield diagonal entries

$$\langle G_{i,*}, G^T_{*,i} \rangle = \langle G_{i,*}, G_{i,*} \rangle = \frac{a_i^2}{a_i^2 + a_j^2} + \frac{a_j^2}{a_i^2 + a_j^2} = \frac{a_i^2 + a_j^2}{a_i^2 + a_j^2} = 1$$

$$\langle G_{j,*}, G^T_{*,j} \rangle = \langle G_{j,*}, G_{j,*} \rangle = \frac{(-a_j)^2}{a_i^2 + a_j^2} + \frac{a_i^2}{a_i^2 + a_j^2} = \frac{a_j^2 + a_i^2}{a_i^2 + a_j^2} = 1$$

and the remaining two inner vector products yield off-diagonal entries

$$\langle G_{i,*}, G^T_{*,j} \rangle = \langle G_{i,*}, G_{j,*} \rangle = \frac{a_i \cdot (-a_j)}{a_i^2 + a_j^2} + \frac{a_j \cdot a_i}{a_i^2 + a_j^2} = \frac{a_i \cdot (-a_j) + a_j \cdot a_i}{a_i^2 + a_j^2} = 0$$

(similarly $\langle G_{j,*}, G^T_{*,i} \rangle = 0$). Hence, $G \cdot G^T = I_m = G \cdot G^{-1}$.

The matrix M in

$$p \cdot \|\mathbf{a}\|_2 \cdot \mathbf{e}_0 = p \cdot M \cdot \mathbf{a} \approx M \cdot \mathbf{y}$$

is built iteratively as

$$M = G_1 \cdot \ldots, G_{m-1}$$

by orthogonalization with respect to the Cartesian basis directions $\mathbf{e}_{m-1}, \ldots, \mathbf{e}_1$.

Note that the product of orthogonal matrices is orthogonal:

$$(G_i \cdot G_j) \cdot (G_i \cdot G_j)^T = G_i \cdot G_j \cdot G_j^T \cdot G_i^T = G_i \cdot G_i^T = I_m$$

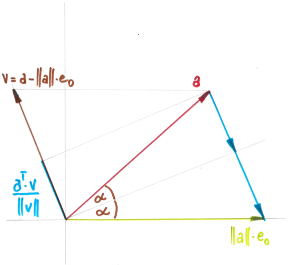
```
template<typename T, int M>
Eigen::Matrix<T,M,1> g(const Eigen::Matrix<T,M,1> &x) { return x; }

template<typename T, int M>
T Givens(Eigen::Matrix<T,M,1> a, Eigen::Matrix<T,M,1> y) {
  auto m=a.size();
  for (auto i=m-2;i>=0;i--) {
    Eigen::Matrix<T,M,M> G=Eigen::Matrix<T,M,M>::Identity(m,m);
    T norm_a_tilde=a.block(i,0,2,1).norm();
    G(i,i)=a(i)/norm_a_tilde; G(i+1,i+1)=G(i,i);
    G(i,i+1)=a(i+1)/norm_a_tilde; G(i+1,i)=-G(i,i+1);
    y=G*y; a=G*a;
  }
  return y(0)/a(0);
}
```

Outline

To transform $\mathbf{a} \in R^n$ into $\|\mathbf{a}\| \cdot \mathbf{e}_0$ twice the vector projection of $\mathbf{a}$ onto $\mathbf{v} = \mathbf{a} - \|\mathbf{a}\| \cdot \mathbf{e}_0$ needs to be subtracted from $\mathbf{a}$.

$$\mathbf{a_v} = \frac{\langle \mathbf{a}, \mathbf{v} \rangle}{\|\mathbf{v}\|} \cdot \frac{\mathbf{v}}{\|\mathbf{v}\|} = \frac{\langle \mathbf{a}, \mathbf{v} \rangle \cdot \mathbf{v}}{\|\mathbf{v}\|^2}$$

$$= \frac{\mathbf{a}^T \cdot \mathbf{v} \cdot \mathbf{v}}{\|\mathbf{v}\|^2} = \frac{\mathbf{v} \cdot \mathbf{a}^T \cdot \mathbf{v}}{\|\mathbf{v}\|^2} = \frac{\mathbf{v} \cdot \mathbf{v}^T \cdot \mathbf{a}}{\|\mathbf{v}\|^2}$$



It follows

$$\|\mathbf{a}\| \cdot \mathbf{e}_0 = \mathbf{a} - 2 \cdot \mathbf{a_v} = \mathbf{a} - 2 \cdot \frac{\mathbf{v} \cdot \mathbf{v}^T \cdot \mathbf{a}}{\|\mathbf{v}\|^2} .$$

From the above follows

$$\|\mathbf{a}\| \cdot \mathbf{e}_0 = \mathbf{a} - 2 \cdot \frac{\mathbf{v} \cdot \mathbf{v}^T \cdot \mathbf{a}}{\|\mathbf{v}\|^2} = \mathbf{a} - 2 \cdot \frac{\mathbf{v} \cdot \mathbf{v}^T}{\langle \mathbf{v}, \mathbf{v} \rangle} \cdot \mathbf{a} = \left(I_m - 2 \cdot \frac{\mathbf{v} \cdot \mathbf{v}^T}{\mathbf{v}^T \cdot \mathbf{v}} \right) \cdot \mathbf{a}$$

Recall that a unit vector (length equal to one) in the direction of a given vector $\mathbf{v} \in R^n$ is obtained by scaling $\mathbf{v}$ with the inverse of its norm, i.e,

$$\mathbf{e}_\mathbf{v} = \frac{\mathbf{v}}{\|\mathbf{v}\|} .$$

The Householder matrix

$$H = I - 2 \cdot \frac{\mathbf{v} \cdot \mathbf{v}^T}{\mathbf{v}^T \cdot \mathbf{v}}$$

is symmetric and orthogonal.

Orthogonality follows from

$$
\begin{aligned}
H^T \cdot H = H \cdot H &= \left(I_m - 2 \cdot \frac{\mathbf{v} \cdot \mathbf{v}^T}{\mathbf{v}^T \cdot \mathbf{v}}\right) \cdot \left(I_m - 2 \cdot \frac{\mathbf{v} \cdot \mathbf{v}^T}{\mathbf{v}^T \cdot \mathbf{v}}\right) \\
&= I_m - 2 \cdot \frac{\mathbf{v} \cdot \mathbf{v}^T}{\mathbf{v}^T \cdot \mathbf{v}} - 2 \cdot \frac{\mathbf{v} \cdot \mathbf{v}^T}{\mathbf{v}^T \cdot \mathbf{v}} + 2 \cdot \frac{\mathbf{v} \cdot \mathbf{v}^T}{\mathbf{v}^T \cdot \mathbf{v}} \cdot 2 \cdot \frac{\mathbf{v} \cdot \mathbf{v}^T}{\mathbf{v}^T \cdot \mathbf{v}} \\
&= I_m - 4 \cdot \frac{\mathbf{v} \cdot \mathbf{v}^T}{\mathbf{v}^T \cdot \mathbf{v}} + 4 \cdot \frac{\mathbf{v} \cdot (\mathbf{v}^T \cdot \mathbf{v}) \cdot \mathbf{v}^T}{(\mathbf{v}^T \cdot \mathbf{v}) \cdot \mathbf{v}^T \cdot \mathbf{v}} \\
&= I_m = H^{-1} \cdot H
\end{aligned}
$$

Implementation


```
template<typename T, int M>
Eigen::Matrix<T,M,1> g(const Eigen::Matrix<T,M,1> &x) { return x; }

template<typename T, int M>
T Householder(Eigen::Matrix<T,M,1> x, Eigen::Matrix<T,M,1> y) {
  using VT=Eigen::Matrix<T,M,1>; using MT=Eigen::Matrix<T,M,M>;
  auto m=x.size();
  VT v=x+x(0)/fabs(x(0))*x.norm()*VT::Unit(m,0);
  MT H=MT::Identity(m,m)-2*v*v.transpose()/v.dot(v);
  x=H*x; y=H*y;
  return y(0)/x(0);
}
```

Outline

Summary

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

Next Steps

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