DATA COMPRESSION

A typical scenario in many sciences is acquisition of m numbers to describe some object that is understood to actually require only $n \ll m$ parameters. For example, m voltage measurements u_i of an alternating current could readily be reduced to three parameters, the amplitude, phase and frequency $u(t) = a\sin(\omega t + \varphi)$. Very often a simple first-degree polynomial approximation $y = ax + b$ is sought for a large data set $D = \{(x_i, y_i), i = 1, \ldots, m\}$. All of these are instances of data compression, a problem that can be solved in a linear algebra framework.

1. Projection

Consider a partition of a vector space U into orthogonal subspaces $U = V \oplus W$, $V = W^{\perp}$, $W = V^{\perp}$. Within the typical scenario described above $U = \mathbb{R}^m$, $V \subset \mathbb{R}^m$, $W \subset \mathbb{R}^m$, $\dim V = n$, $\dim W = m - n$. If $\boldsymbol{V} = [\ \boldsymbol{v}_1\ \ \ldots\ \ \boldsymbol{v}_n\ ] \in \mathbb{R}^{m\times n}$ is a basis for V and $\boldsymbol{W} = [\ \boldsymbol{w}_1\ \ \ldots\ \ \boldsymbol{w}_{m-n}\ ] \in \mathbb{R}^{m\times(m-n)}$ is a basis for W, then $\boldsymbol{U} = [\ \boldsymbol{v}_1\ \ \ldots\ \ \boldsymbol{v}_n\ \ \boldsymbol{w}_1\ \ \ldots\ \ \boldsymbol{w}_{m-n}\ ]$ is a basis for U. Even though the matrices $\boldsymbol{V}, \boldsymbol{W}$ are not necessarily square, they are said to be orthogonal, in the sense that all columns are of unit norm and orthogonal to one another. Computation of the matrix product $\boldsymbol{V}^T\boldsymbol{V}$ leads to the formation of the identity matrix within $\mathbb{R}^n$

$$\boldsymbol{V}^T\boldsymbol{V} = \begin{bmatrix} \boldsymbol{v}_1^T \\ \boldsymbol{v}_2^T \\ \vdots \\ \boldsymbol{v}_n^T \end{bmatrix} [\ \boldsymbol{v}_1\ \ \boldsymbol{v}_2\ \ \ldots\ \ \boldsymbol{v}_n\ ] = \begin{bmatrix} \boldsymbol{v}_1^T\boldsymbol{v}_1 & \boldsymbol{v}_1^T\boldsymbol{v}_2 & \ldots & \boldsymbol{v}_1^T\boldsymbol{v}_n \\ \boldsymbol{v}_2^T\boldsymbol{v}_1 & \boldsymbol{v}_2^T\boldsymbol{v}_2 & \ldots & \boldsymbol{v}_2^T\boldsymbol{v}_n \\ \vdots & \vdots & \ddots & \vdots \\ \boldsymbol{v}_n^T\boldsymbol{v}_1 & \boldsymbol{v}_n^T\boldsymbol{v}_2 & \ldots & \boldsymbol{v}_n^T\boldsymbol{v}_n \end{bmatrix} = \boldsymbol{I}_n.$$

Similarly, $\boldsymbol{W}^T\boldsymbol{W} = \boldsymbol{I}_{m-n}$. Whereas for the square orthogonal matrix $\boldsymbol{U}$ multiplication both on the left and the right by its transpose leads to the formation of the identity matrix

$$\boldsymbol{U}^T\boldsymbol{U} = \boldsymbol{U}\boldsymbol{U}^T = \boldsymbol{I}_m,$$

the same operations applied to rectangular orthogonal matrices lead to different results

$$\boldsymbol{V}^T\boldsymbol{V} = \boldsymbol{I}_n, \boldsymbol{V}\boldsymbol{V}^T = [\ \boldsymbol{v}_1\ \ \boldsymbol{v}_2\ \ \ldots\ \ \boldsymbol{v}_n\ ] \begin{bmatrix} \boldsymbol{v}_1^T \\ \boldsymbol{v}_2^T \\ \vdots \\ \boldsymbol{v}_n^T \end{bmatrix} = \sum_{i=1}^{n} \boldsymbol{v}_i\boldsymbol{v}_i^T, \operatorname{rank}(\boldsymbol{v}_i\boldsymbol{v}_i^T) = 1$$

A simple example is provided by taking $\boldsymbol{V} = \boldsymbol{I}_{m,n}$, the first n columns of the identity matrix in which case

$$\boldsymbol{V}\boldsymbol{V}^T = \sum_{i=1}^{n} \boldsymbol{e}_i\boldsymbol{e}_i^T = \begin{bmatrix} \boldsymbol{I}_n & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{0} \end{bmatrix} \in \mathbb{R}^{m\times m}.$$

Applying $\boldsymbol{P} = \boldsymbol{V}\boldsymbol{V}^T$ to some vector $\boldsymbol{b} \in \mathbb{R}^m$ leads to a vector $\boldsymbol{r} = \boldsymbol{P}\boldsymbol{b}$ whose first n components are those of $\boldsymbol{b}$, and the remaining $m - n$ are zero. The subtraction $\boldsymbol{b} - \boldsymbol{r}$ leads to a new vector $\boldsymbol{s} = (\boldsymbol{I} - \boldsymbol{P})\boldsymbol{b}$ that has the first components equal to zero, and the remaining $m - n$ the same as those of $\boldsymbol{b}$. Such operations are referred to as *projections*, and for $\boldsymbol{V} = \boldsymbol{I}_{m,n}$ correspond to projection onto the span$\{\boldsymbol{e}_1, \ldots, \boldsymbol{e}_n\}$.

```
octave] I4=eye(5); V=I4(:,1:2); P=V*V';␣Q=I4-P;
        b=rand(5,1); r=P*b; s=Q*b; disp([P b r s])
   1.00000   0.00000   0.00000   0.00000   0.00000   0.42253   0.42253
0.00000
   0.00000   1.00000   0.00000   0.00000   0.00000   0.95900   0.95900
0.00000
   0.00000   0.00000   0.00000   0.00000   0.00000   0.41781   0.00000
0.41781
   0.00000   0.00000   0.00000   0.00000   0.00000   0.45744   0.00000
0.45744
   0.00000   0.00000   0.00000   0.00000   0.00000   0.49784   0.00000
0.49784
```

```
octave]
```

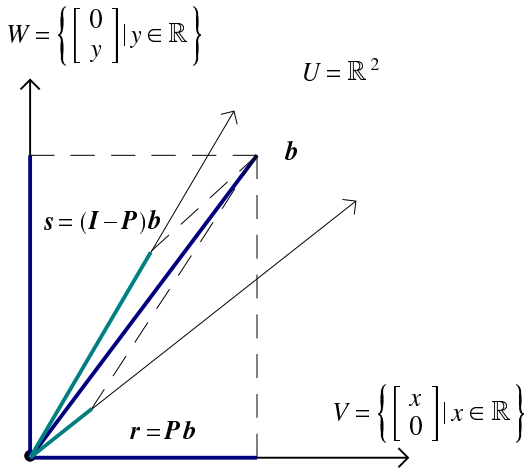


Figure 1. Projection in $\mathbb{R}^2$. The vectors $\boldsymbol{r}, \boldsymbol{s} \in \mathbb{R}^2$ have two components, but could be expressed through scaling of $\boldsymbol{e}_1, \boldsymbol{e}_2$.

Returning to the general case, the orthogonal matrices $\boldsymbol{U} \in \mathbb{R}^{m \times m}$, $\boldsymbol{V} \in \mathbb{R}^{m \times n}$, $\boldsymbol{W} \in \mathbb{R}^{m \times (m-n)}$ are associated with linear mappings $\boldsymbol{b} = \boldsymbol{f}(\boldsymbol{x}) = \boldsymbol{U}\boldsymbol{x}$, $\boldsymbol{r} = \boldsymbol{g}(\boldsymbol{b}) = \boldsymbol{P}\boldsymbol{b}$, $\boldsymbol{s} = \boldsymbol{h}(\boldsymbol{b}) = (\boldsymbol{I} - \boldsymbol{P})\,\boldsymbol{b}$. The mapping $\boldsymbol{f}$ gives the components in the $\boldsymbol{I}$ basis of a vector whose components in the $\boldsymbol{U}$ basis are $\boldsymbol{x}$. The mappings $\boldsymbol{g}, \boldsymbol{h}$ project a vector onto span$\{\boldsymbol{v}_1, \ldots, \boldsymbol{v}_n\}$, span$\{\boldsymbol{w}_1, \ldots, \boldsymbol{w}_{m-n}\}$, respectively. When $\boldsymbol{V}, \boldsymbol{W}$ are orthogonal matrices the projections are also orthogonal $\boldsymbol{r} \perp \boldsymbol{s}$. Projection can also be carried out onto nonorthogonal spanning sets, but the process is fraught with possible error, especially when the angle between basis vectors is small, and will be avoided henceforth.

Notice that projection of a vector already in the spanning set simply returns the same vector, which leads to a general definition.

DEFINITION. *The mapping is called a projection if* $\boldsymbol{f} \circ \boldsymbol{f} = \boldsymbol{f}$, *or if for any* $\boldsymbol{u} \in U$, $\boldsymbol{f}(\boldsymbol{f}(\boldsymbol{u})) = \boldsymbol{f}(\boldsymbol{u})$. *With* $\boldsymbol{P}$ *the matrix associated* $\boldsymbol{f}$, *a projection matrix satisfies* $\boldsymbol{P}^2 = \boldsymbol{P}$.

$$\boldsymbol{P} = \boldsymbol{V}\boldsymbol{V}^T$$

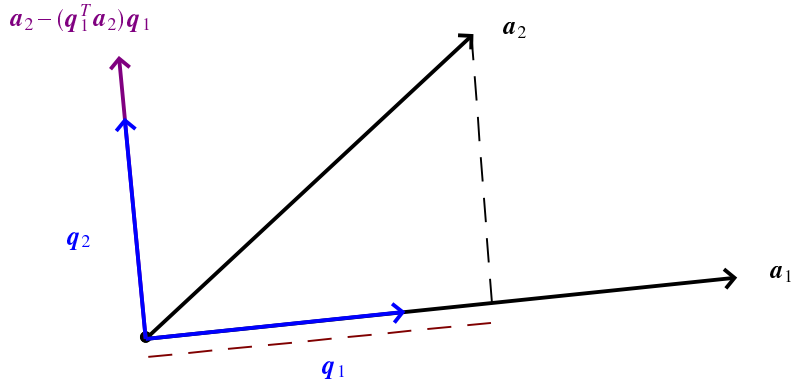
$$\boldsymbol{P}^2 = \boldsymbol{P}\boldsymbol{P} = \boldsymbol{V}\boldsymbol{V}^T\boldsymbol{V}\boldsymbol{V}^T = \boldsymbol{V}\,(\boldsymbol{V}^T\boldsymbol{V})\,\boldsymbol{V}^T = \boldsymbol{V}\boldsymbol{I}\boldsymbol{V}^T = \boldsymbol{V}\boldsymbol{V}^T = \boldsymbol{P}$$

2. Gram-Schmidt

Orthonormal vector sets $\{\boldsymbol{q}_1, \ldots, \boldsymbol{q}_n\}$ are of the greatest practical utility, leading to the question of whether some such a set can be obtained from an arbitrary set of vectors $\{\boldsymbol{a}_1, \ldots, \boldsymbol{a}_n\}$. This is possible for independent vectors, through what is known as the Gram-Schmidt algorithm

1. Start with an arbitrary direction $\boldsymbol{a}_1$
2. Divide by its norm to obtain a unit-norm vector $\boldsymbol{q}_1 = \boldsymbol{a}_1 / \|\boldsymbol{a}_1\|$
3. Choose another direction $\boldsymbol{a}_2$
4. Subtract off its component along previous direction(s) $\boldsymbol{a}_2 - (\boldsymbol{q}_1^T \boldsymbol{a}_2)\boldsymbol{q}_1$

5. Divide by norm $\boldsymbol{q}_2 = (\boldsymbol{a}_2 - (\boldsymbol{q}_1^T \boldsymbol{a}_2)\boldsymbol{q}_1) / \|\boldsymbol{a}_2 - (\boldsymbol{q}_1^T \boldsymbol{a}_2)\boldsymbol{q}_1\|$
6. Repeat the above



$$\boldsymbol{P}_1 \boldsymbol{a}_2 = (\boldsymbol{q}_1 \boldsymbol{q}_1^T)\boldsymbol{a}_2 = \boldsymbol{q}_1 (\boldsymbol{q}_1^T \boldsymbol{a}_2) = (\boldsymbol{q}_1^T \boldsymbol{a}_2)\, \boldsymbol{q}_1$$

The above geometrical description can be expressed in terms of matrix operations as

$$\boldsymbol{A} = (\, \boldsymbol{a}_1 \;\; \boldsymbol{a}_2 \;\; \ldots \;\; \boldsymbol{a}_n \,) = (\, \boldsymbol{q}_1 \;\; \boldsymbol{q}_2 \;\; \ldots \;\; \boldsymbol{q}_n \,) \begin{pmatrix} r_{11} & r_{12} & r_{13} & \ldots & r_{1n} \\ 0 & r_{22} & r_{23} & \ldots & r_{2n} \\ 0 & 0 & r_{33} & \ldots & r_{3n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \ldots & \ldots & r_{mn} \end{pmatrix} = \boldsymbol{Q}\boldsymbol{R},$$

equivalent to the system

$$\begin{cases} \boldsymbol{a}_1 = r_{11}\boldsymbol{q}_1 \\ \boldsymbol{a}_2 = r_{12}\boldsymbol{q}_1 + r_{22}\boldsymbol{q}_2 \\ \vdots \\ \boldsymbol{a}_n = r_{1n}\boldsymbol{q}_1 + r_{2n}\boldsymbol{q}_2 + \ldots + r_{nn}\boldsymbol{q}_n \end{cases}.$$

The system is easily solved by *forward substitution* resulting in what is known as the (modified) *Gram-Schmidt algorithm*, transcribed below both in pseudo-code and in Octave.

Algorithm (Gram-Schmidt)

Given n vectors $\boldsymbol{a}_1, \ldots, \boldsymbol{a}_n$
Initialize $\boldsymbol{q}_1 = \boldsymbol{a}_1, .., \boldsymbol{q}_n = \boldsymbol{a}_n$, $\boldsymbol{R} = \boldsymbol{I}_n$
for $i = 1$ to n
  $r_{ii} = (\boldsymbol{q}_i^T \boldsymbol{q}_i)^{1/2}$
  if $r_{ii} < \epsilon$ break;
  $\boldsymbol{q}_i = \boldsymbol{q}_i / r_{ii}$
  for $j = i+1$ to n
    $r_{ij} = \boldsymbol{q}_i^T \boldsymbol{a}_j$; $\boldsymbol{q}_j = \boldsymbol{q}_j - r_{ij}\boldsymbol{q}_i$
  end
end
return $\boldsymbol{Q}, \boldsymbol{R}$

```
octave] function [Q,R] = mgs(A)
          [m,n]=size(A); Q=A; R=eye(n);
          for i=1:n
            R(i,i) = sqrt(Q(:,i)'*Q(:,i));
            if (R(i,i)<eps) break;
            Q(:,i) = Q(:,i)/R(i,i);
            for j=i+1:n
              R(i,j) = Q(:,i)'*A(:,j);
              Q(:,j) = Q(:,j) - R(i,j)*Q(:,i);
            end;
          end;
        end
octave]
```

Note that the normalization condition $\|\boldsymbol{q}_{ii}\| = 1$ is satisifed by two values $\pm r_{ii}$, so results from the above implementation might give orthogonal vectors $\boldsymbol{q}_1, \ldots, \boldsymbol{q}_n$ of different orientations than those returned by the Octave `qr` function. The implementation provided by computational packages such as Octave contain many refinements of the basic algorithm and it's usually preferable to use these in applications.

```
octave] A=rand(4); [Q,R]=mgs(A); disp([Q R])
   0.82757  -0.25921  -0.49326   0.06802   0.83553   0.64827   1.24651
1.05301
   0.19408   0.53127   0.15805   0.80939   0.00000   0.93177   0.82700
0.87551
   0.22006   0.79553  -0.12477  -0.55058   0.00000   0.00000   0.38433
-0.20336
   0.47857  -0.13302   0.84625  -0.19270   0.00000   0.00000   0.00000
0.42469
octave] [Q1,R1]=qr(A); disp([Q1 R1])
  -0.82757   0.25921  -0.49326  -0.06802  -0.83553  -0.64827  -1.24651
-1.05301
  -0.19408  -0.53127   0.15805  -0.80939   0.00000  -0.93177  -0.82700
-0.87551
  -0.22006  -0.79553  -0.12477   0.55058   0.00000   0.00000   0.38433
-0.20336
  -0.47857   0.13302   0.84625   0.19270   0.00000   0.00000   0.00000
-0.42469
octave] disp([norm(A-Q*R) norm(A-Q1*R1)])
   1.1102e-16   8.0390e-16
octave]
```

By analogy to arithmetic and polynomial algebra, the Gram-Schmidt algorithm furnishes a *factorization*

$$\boldsymbol{QR} = \boldsymbol{A}$$

with $\boldsymbol{Q} \in \mathbb{R}^{m \times n}$ with orthonormal columns and $\boldsymbol{R} \in \mathbb{R}^{n \times n}$ an upper triangular matrix, known as the QR-factorization. Since the column vectors within $\boldsymbol{Q}$ were obtained through linear combinations of the column vectors of $\boldsymbol{A}$ we have

$$C(\boldsymbol{A}) = C(\boldsymbol{Q}) \neq C(\boldsymbol{R})$$

$$\boldsymbol{AX} = \boldsymbol{B}, \boldsymbol{A}[\, \boldsymbol{x}_1 \ \ldots \ \boldsymbol{x}_n \,] = [\, \boldsymbol{Ax}_1 \ \ldots \ \boldsymbol{Ax}_n \,].$$

The QR-factorization can be used to solve basic problems within linear algebra.

```
octave] A=[3 2; 1 2]
A =

   3   2
   1   2

octave] [Q R]=qr(A)
Q =

  -0.94868  -0.31623
  -0.31623   0.94868

R =

  -3.16228  -2.52982
   0.00000   1.26491

octave]
```

3. *QR* solution of linear algebra problems

3.1. Transformation of coordinates

Recall that when given a vector $\boldsymbol{b} \in \mathbb{R}^m$, an implicit basis is assumed, the canonical basis given by the column vectors of the identity matrix $\boldsymbol{I} \in \mathbb{R}^{m \times m}$. The coordinates $\boldsymbol{x}$ in another basis $\boldsymbol{A} \in \mathbb{R}^{m \times m}$ can be found by solving the equation

$$\boldsymbol{I}\boldsymbol{b} = \boldsymbol{b} = \boldsymbol{A}\boldsymbol{x},$$

by an intermediate change of coordinates to the orthogonal basis $\boldsymbol{Q}$. Since the basis $\boldsymbol{Q}$ is orthogonal the relation $\boldsymbol{Q}^T\boldsymbol{Q} = \boldsymbol{I}$ holds, and changes of coordinates from $\boldsymbol{I}$ to $\boldsymbol{Q}$, $\boldsymbol{Q}\boldsymbol{c} = \boldsymbol{b}$, are easily computed $\boldsymbol{c} = \boldsymbol{Q}^T\boldsymbol{b}$. Since matrix multiplication is associative

$$\boldsymbol{b} = \boldsymbol{A}\boldsymbol{x} = (\boldsymbol{Q}\boldsymbol{R})\boldsymbol{x} = \boldsymbol{Q}(\boldsymbol{R}\boldsymbol{x}),$$

the relations $\boldsymbol{R}\boldsymbol{x} = \boldsymbol{Q}^T\boldsymbol{b} = \boldsymbol{c}$ are obtained, stating that $\boldsymbol{x}$ also contains the coordinates of $\boldsymbol{c}$ in the basis $\boldsymbol{R}$. The three steps are:

1. Compute the QR-factorization, $\boldsymbol{Q}\boldsymbol{R} = \boldsymbol{A}$;
2. Find the coordinates of $\boldsymbol{b}$ in the orthogonal basis $\boldsymbol{Q}$, $\boldsymbol{c} = \boldsymbol{Q}^T\boldsymbol{b}$;
3. Find the coordinates of $\boldsymbol{x}$ in basis $\boldsymbol{R}$, $\boldsymbol{R}\boldsymbol{x} = \boldsymbol{c}$.

Since $\boldsymbol{R}$ is upper-triangular,

$$\begin{pmatrix} r_{11} & r_{12} & r_{13} & \cdots & r_{1m} \\ 0 & r_{22} & r_{23} & \cdots & r_{2m} \\ 0 & 0 & r_{33} & \cdots & r_{3m} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \cdots & r_{mm} \end{pmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_{m-1} \\ x_m \end{bmatrix} = \begin{bmatrix} c_1 \\ c_2 \\ \vdots \\ c_{m-1} \\ c_m \end{bmatrix}$$

the coordinates of $\boldsymbol{c}$ in the $\boldsymbol{R}$ basis are easily found by *back substitution*.

Algorithm (Back substitution)

Given R upper-triangular, vectors $\boldsymbol{c}$
for $i = m$ down to 1
  if $r_{ii} < \epsilon$ break;
  $x_i = c_i / r_{ii}$
  for $j = i$-1 down to 1
    $c_j = c_j - r_{ji} x_i$
  end
end
return $\boldsymbol{x}$

```
octave] function x=bcks(R,c)
          [m,n]=size(R); x=zeros(m,1);
          for i=m:-1:1
            x(i) = c(i)/R(i,i);
            for j=i-1:-1:1
              c(j) = c(j) - R(j,i)*x(i);
            end;
          end;
        end
octave]
```

The above operations are carried out in the background by the Octave backslash operation `A\b` to solve `A*x=b`, inspired by the scalar mnemonic $ax = b \Rightarrow x = (1/a)\, b$. Again, many additional refinements of the basic algorithm argue for using the built-in Octave functions, even though the above implementations can be verified as correct.

```
octave] xex=rand(4,1); b=A*xex; [Q,R]=mgs(A); c=Q'*b; x=bcks(R,c); xO=A\b;
octave] disp([xex x xO])
   0.96838   0.96838   0.96838
   0.31829   0.31829   0.31829
   0.58529   0.58529   0.58529
   0.38250   0.38250   0.38250
octave]
```

3.2. General orthogonal bases

The above approch for the real vector space $\mathscr{R}_m$ can be used to determine orthogonal bases for any other vector space by appropriate modification of the scalar product. For example, within the space of smooth functions $\mathscr{C}^\infty[-1,1]$ that can differentiated an arbitrary number of times, the Taylor series

$$f(x) = f(0) \cdot 1 + f'(0) \cdot x + \tfrac{1}{2} f''(0) \cdot x^2 + \cdots + \tfrac{1}{n!} f^{(n)}(0) \cdot x^n + \cdots +$$

is seen to be a linear combination of the monomial basis $\boldsymbol{M} = [\, 1 \;\; x \;\; x^2 \;\; \ldots \,]$ with scaling coefficients $\left\{ f(0), f'(0), \tfrac{1}{2} f''(0), \ldots \right\}$. The scalar product

$$(f, g) = \int_{-1}^{1} f(x)\, g(x)\, \mathrm{d}x$$

can be seen as the extension to the $[-1,1]$ continuum of a the vector dot product. Orthogonalization of the monomial basis with the above scalar product leads to the definition of another family of polynomials, known as the Legendre polynomials

$$Q_0(x) = \left(\tfrac{1}{2}\right)^{1/2} \cdot 1,\; Q_1(x) = \left(\tfrac{3}{2}\right)^{1/2} \cdot x,\; Q_2(x) = \left(\tfrac{5}{8}\right)^{1/2} \cdot (3x^2 - 1),\; Q_4(x) = \left(\tfrac{7}{8}\right)^{1/2} \cdot (5x^3 - 3x), \ldots .$$

The Legendre polynomials are usually given with a different scaling such that $P_k(1) = 1$, rather than the unit norm condition $\|Q_k\| = (Q_k, Q_k)^{1/2} = 1$. The above results can be recovered by sampling of the interval $[-1,1]$ at points $x_i = (i-1)\, h - 1$, $h = 2/(m-1)$, $i = 1, \ldots, m$, by approximation of the integral by a Riemann sum

$$\int_{-1}^{1} f(x)\, L_j(x)\, \mathrm{d}x \cong h \sum_{i=1}^{m} f(x_i) L_j(x_i) = h \boldsymbol{f}^T \boldsymbol{L}_j .$$

```
octave] m=50; h=2/(m-1); x=(-1:h:1)';  M=[x.^0  x.^1  x.^2  x.^3  x.^4];
        [Q,R]=mgs(M);
        S=diag(1./Q(m,:)); P=Q*S; sc=[-1 1 -1 1];
        figure(1);  plot(x,M(:,1),x,M(:,2),x,M(:,3),x,M(:,4));  axis(sc);
        grid on;
        figure(2);  plot(x,P(:,1),x,P(:,2),x,P(:,3),x,P(:,4));  axis(sc);
        grid on;
octave]
```

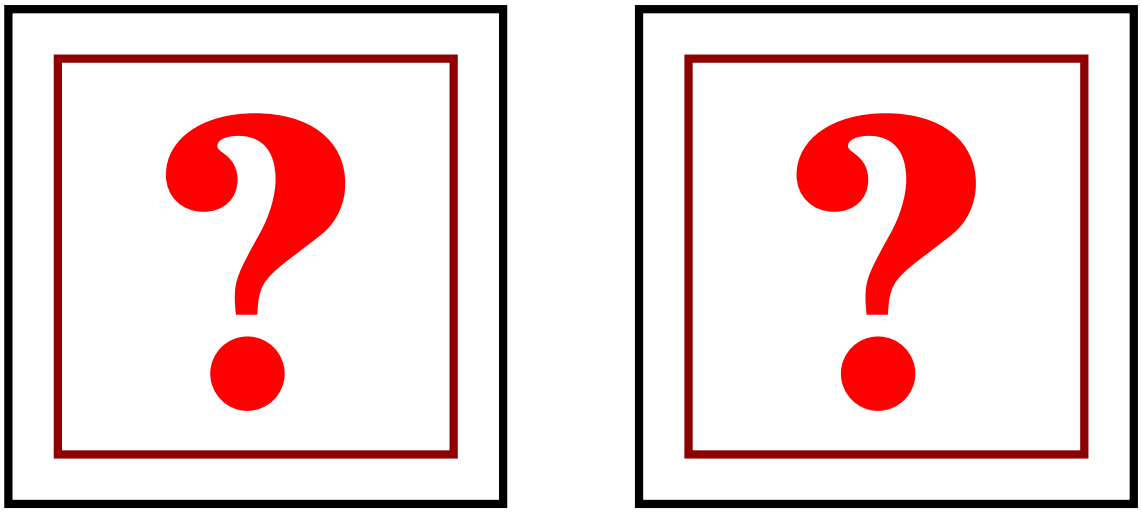


Figure 2. Comparison of monomial basis (left) to Legendre polynomial basis (right). The "resolution" of $P_3(x)$ can be interpreted as the number of crossings of the $y=0$ ordinate axis, and is greater than that of the corresponding monomial x^3.

3.3. Least squares

The approach to compressing data $D=\{(x_i,y_i)|\,i=1,\ldots,m\}$ suggested by calculus concepts is to form the sum of squared differences between $y(x_i)$ and y_i, for example for $y(x)=a_0+a_1x$ when carrying out linear regression,

$$S(a_0,a_1)=\sum_{i=1}^{m}\,(y(x_i)-y_i)^2=\sum_{i=1}^{m}\,(a_0+a_1x_i-y_i)^2$$

and seek (a_0,a_1) that minimize $S(a_0,a_1)$. The function $S(a_0,a_1)\geqslant 0$ can be thought of as the height of a surface above the a_0a_1 plane, and the gradient ∇S is defined as a vector in the direction of steepest slope. When at some point on the surface if the gradient is different from the zero vector $\nabla S\neq\mathbf{0}$, travel in the direction of the gradient would increase the height, and travel in the opposite direction would decrease the height. The minimal value of S would be attained when no local travel could decrease the function value, which is known as stationarity condition, stated as $\nabla S=0$. Applying this to determining the coefficients (a_0,a_1) of a linear regression leads to the equations

$$\frac{\partial S}{\partial a_0}=0\Rightarrow 2\sum_{i=1}^{m}\,(a_0+a_1x_i-y_i)=0\Leftrightarrow ma_0+\left(\sum_{i=1}^{m}\,x_i\right)a_1=\sum_{i=1}^{m}\,y_i,$$

$$\frac{\partial S}{\partial a_1}=0\Rightarrow 2\sum_{i=1}^{m}\,(a_0+a_1x_i-y_i)x_i=0\Leftrightarrow \left(\sum_{i=1}^{m}\,x_i\right)a_0+\left(\sum_{i=1}^{m}\,x_i^2\right)a_1=\sum_{i=1}^{m}\,x_iy_i.$$

The above calculations can become tedious, and do not illuminate the geometrical essence of the calculation, which can be brought out by reformulation in terms of a matrix-vector product that highlights the particular linear combination that is sought in a liner regression. Form a vector of errors with components $e_i = y(x_i) - y_i$, which for linear regression is $y(x) = a_0 + a_1 x$. Recognize that $y(x_i)$ is a linear combination of 1 and x_i with coefficients a_0, a_1, or in vector form

$$\boldsymbol{e} = \begin{pmatrix} 1 & x_1 \\ \vdots & \vdots \\ 1 & x_m \end{pmatrix} \begin{pmatrix} a_0 \\ a_1 \end{pmatrix} - \boldsymbol{y} = (\ \mathbf{1}\ \ \boldsymbol{x}\ )\boldsymbol{a} - \boldsymbol{y} = \boldsymbol{A}\boldsymbol{a} - \boldsymbol{y}$$

The norm of the error vector $\|\boldsymbol{e}\|$ is smallest when $\boldsymbol{A}\boldsymbol{a}$ is as close as possible to $\boldsymbol{y}$. Since $\boldsymbol{A}\boldsymbol{a}$ is within the column space of $C(\boldsymbol{A})$, $\boldsymbol{A}\boldsymbol{a} \in C(\boldsymbol{A})$, the required condition is for $\boldsymbol{e}$ to be orthogonal to the column space

$$\boldsymbol{e} \perp C(\boldsymbol{A}) \Rightarrow \boldsymbol{A}^T\boldsymbol{e} = \begin{pmatrix} \mathbf{1}^T \\ \boldsymbol{x}^T \end{pmatrix} \boldsymbol{e} = \begin{pmatrix} \mathbf{1}^T\boldsymbol{e} \\ \boldsymbol{x}^T\boldsymbol{e} \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix} = \mathbf{0}$$

$$\boldsymbol{A}^T\boldsymbol{e} = \mathbf{0} \Leftrightarrow \boldsymbol{A}^T(\boldsymbol{A}\boldsymbol{a} - \boldsymbol{y}) = 0 \Leftrightarrow (\boldsymbol{A}^T\boldsymbol{A})\boldsymbol{a} = \boldsymbol{A}^T\boldsymbol{y} = \boldsymbol{b}.$$

The above is known as the normal system, with $\boldsymbol{N} = \boldsymbol{A}^T\boldsymbol{A}$ is the normal matrix. The system $\boldsymbol{N}\boldsymbol{a} = \boldsymbol{b}$ can be interpreted as seeking the coordinates in the $\boldsymbol{N} = \boldsymbol{A}^T\boldsymbol{A}$ basis of the vector $\boldsymbol{b} = \boldsymbol{A}^T\boldsymbol{y}$. An example can be constructed by randomly perturbing a known function $y(x) = a_0 + a_1 x$ to simulate measurement noise and compare to the approximate $\tilde{\boldsymbol{a}}$ obtained by solving the normal system.

1. Generate some data on a line and perturb it by some random quantities

```
octave] m=100; x=(0:m-1)/m; a=[2; 3];
        a0=a(1); a1=a(2); yex=a0+a1*x; y=(yex+rand(1,m)-0.5)';
octave]
```

2. Form the matrices $\boldsymbol{A}$, $\boldsymbol{N} = \boldsymbol{A}^T\boldsymbol{A}$, vector $\boldsymbol{b} = \boldsymbol{A}^T\boldsymbol{y}$

```
octave] A=ones(m,2); A(:,2)=x(:); N=A'*A; b=A'*y;
octave]
```

3. Solve the system $\boldsymbol{N}\boldsymbol{a} = \boldsymbol{b}$, and form the linear combination $\tilde{\boldsymbol{y}} = \boldsymbol{A}\boldsymbol{a}$ closest to $\boldsymbol{y}$

```
octave] atilde=N\b; disp([a atilde]);
   2.0000   2.0302
   3.0000   2.9628
octave]
```

The normal matrix basis $\boldsymbol{N} = \boldsymbol{A}^T\boldsymbol{A}$ can however be an ill-advised choice. Consider $\boldsymbol{A} \in \mathbb{R}^{2\times 2}$ given by

$$\boldsymbol{A} = [\ \boldsymbol{a}_1\ \ \boldsymbol{a}_2\ ] = \begin{bmatrix} 1 & \cos\theta \\ 0 & \sin\theta \end{bmatrix},$$

where the first column vector is taken from the identity matrix $\boldsymbol{a}_1 = \boldsymbol{e}_1$, and second is the one obtained by rotating it with angle θ. If $\theta = \pi/2$, the normal matrix is orthogonal, $\boldsymbol{A}^T\boldsymbol{A} = \boldsymbol{I}$, but for small θ, $\boldsymbol{A}$ and $\boldsymbol{N} = \boldsymbol{A}^T\boldsymbol{A}$ are approximated as

$$\boldsymbol{A} \cong \begin{bmatrix} 1 & 1 \\ 0 & \theta \end{bmatrix}, \boldsymbol{N} = [\ \boldsymbol{n}_1\ \ \boldsymbol{n}_2\ ] = \begin{bmatrix} 1 & 1 \\ 0 & \theta^2 \end{bmatrix}.$$

When θ is small $\boldsymbol{a}_1, \boldsymbol{a}_2$ are almost colinear, and $\boldsymbol{n}_1, \boldsymbol{n}_2$ even more so. This can lead to amplification of small errors, but can be avoided by recognizing that the best approximation in the 2-norm is identical to the Euclidean concept of orthogonal projection. The orthogonal projector onto $C(\boldsymbol{A})$ is readily found by QR-factorization, and the steps to solve least squares become

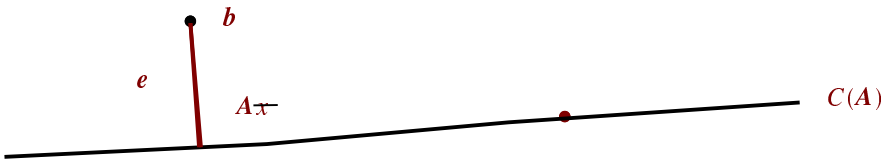
1. Compute $\boldsymbol{Q}\boldsymbol{R} = \boldsymbol{A}$

2. The projection of $\boldsymbol{y}$ onto the column space of $\boldsymbol{A}$ is $\boldsymbol{z} = \boldsymbol{Q}\boldsymbol{Q}^T\boldsymbol{y}$, and has coordinates $\boldsymbol{c} = \boldsymbol{Q}^T\boldsymbol{y}$ in the orthogonal basis $\boldsymbol{Q}$.
3. The same $\boldsymbol{z}$ can also obtained by linear combination of the columns of $\boldsymbol{A}$, $\boldsymbol{z} = \boldsymbol{A}\boldsymbol{a} = \boldsymbol{Q}\boldsymbol{Q}^T\boldsymbol{y}$, and replacing $\boldsymbol{A}$ with its QR-factorization gives $\boldsymbol{Q}\boldsymbol{R}\boldsymbol{a} = \boldsymbol{Q}\boldsymbol{c}$, that leads to the system $\boldsymbol{R}\boldsymbol{a} = \boldsymbol{c}$, solved by back-substitution.

```
octave] [Q,R]=qr(A); c=Q'*y; aQR=R\c; disp([a atilde aQR])
    2.0000    2.0302    2.0302
    3.0000    2.9628    2.9628
octave]
```

The above procedure carried over to approximation by higher degree polynomials.

```
octave] m=100; n=6; x=(0:m-1)/m; x=x'; a=randi(10,n,1); A=[];
        for j=1:n
          A = [A x.^(j-1)];
        end;
        yex=A*a; y=yex+(rand(m,1)-0.5);
octave] N=A'*A; b=A'*y; atilde=inv(N)*b;
        [Q,R]=qr(A); c=Q'*y; aQR=R\c;
        disp([a atilde aQR]);
    8.0000    8.0847    8.0847
    8.0000    7.1480    7.1480
    4.0000    4.2264    4.2264
    4.0000    8.7568    8.7568
   10.0000    2.7420    2.7420
    6.0000    9.0386    9.0386
octave]
```



Given data $\boldsymbol{b}$, form $\boldsymbol{A}$, find $\boldsymbol{x}$, such that $\|\boldsymbol{e}\| = \|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b}\|$ is minimized

$$\boldsymbol{e} = \boldsymbol{b} - \boldsymbol{A}\boldsymbol{x}$$