L. Vandenberghe
ECE133A 5/15/2021

Homework 5 solutions

1. *Exercise T12.12.*

   (a) The sum of the squared distances is

$$\begin{aligned}
&\|p_{i_1} - p_{j_1}\|^2 + \cdots + \|p_{i_L} - p_{j_L}\|^2 \\
&= \left\| \left[\begin{array}{c} u_{i_1} - u_{j_1} \\ v_{i_1} - v_{j_1} \end{array} \right] \right\|^2 + \cdots + \left\| \left[\begin{array}{c} u_{i_L} - u_{j_L} \\ v_{i_L} - v_{j_L} \end{array} \right] \right\|^2 \\
&= (u_{i_1} - u_{j_1})^2 + \cdots + (u_{i_L} - u_{j_L})^2 + (v_{i_1} - v_{j_1})^2 + \cdots + (v_{i_L} - v_{j_L})^2 \\
&= \mathcal{D}(u) + \mathcal{D}(v).
\end{aligned}$$

   (b) We have $\mathcal{D}(u) + \mathcal{D}(v) = \|B^T u\|^2 + \|B^T v\|^2$ where B is the $N \times L$ node incidence matrix

$$B_{ij} = \begin{cases} 1 & \text{edge } j \text{ points to node } i \\ -1 & \text{edge } j \text{ points from node } i \\ 0 & \text{otherwise.} \end{cases}$$

   Suppose we partition B as

$$B = \left[\begin{array}{c} B_{\mathrm{m}} \\ B_{\mathrm{f}} \end{array} \right], \qquad B_{\mathrm{m}} = B_{1:(N-K),1:L}, \qquad B_{\mathrm{f}} = B_{(N-K+1):N,1:L},$$

   and the vectors u and v as $u = (u_{\mathrm{m}}, u_{\mathrm{f}})$ and $v = (v_{\mathrm{m}}, v_{\mathrm{f}})$ where

$$u_{\mathrm{m}} = u_{1:(N-K)}, \qquad v_{\mathrm{m}} = v_{1:(N-K)}, \qquad u_{\mathrm{f}} = u_{(N-K+1):N}, \qquad v_{\mathrm{f}} = v_{(N-K+1):N}.$$

   Then the objective function can be expressed as

$$\begin{aligned}
\left\|B^T u\right\|^2 + \left\|B^T v\right\|^2 &= \left\|B_{\mathrm{m}}^T u_{\mathrm{m}} + B_{\mathrm{f}}^T u_{\mathrm{f}}\right\|^2 + \left\|B_{\mathrm{m}}^T v_{\mathrm{m}} + B_{\mathrm{f}}^T v_{\mathrm{f}}\right\|^2 \\
&= \left\| \left[\begin{array}{cc} B_{\mathrm{m}}^T & 0 \\ 0 & B_{\mathrm{m}}^T \end{array} \right] \left[\begin{array}{c} u_{\mathrm{m}} \\ v_{\mathrm{m}} \end{array} \right] + \left[\begin{array}{c} B_{\mathrm{f}}^T u_{\mathrm{f}} \\ B_{\mathrm{f}}^T v_{\mathrm{f}} \end{array} \right] \right\|^2.
\end{aligned}$$

   Minimizing this is a least squares problem with variable $x = (u_{\mathrm{m}}, v_{\mathrm{m}})$,

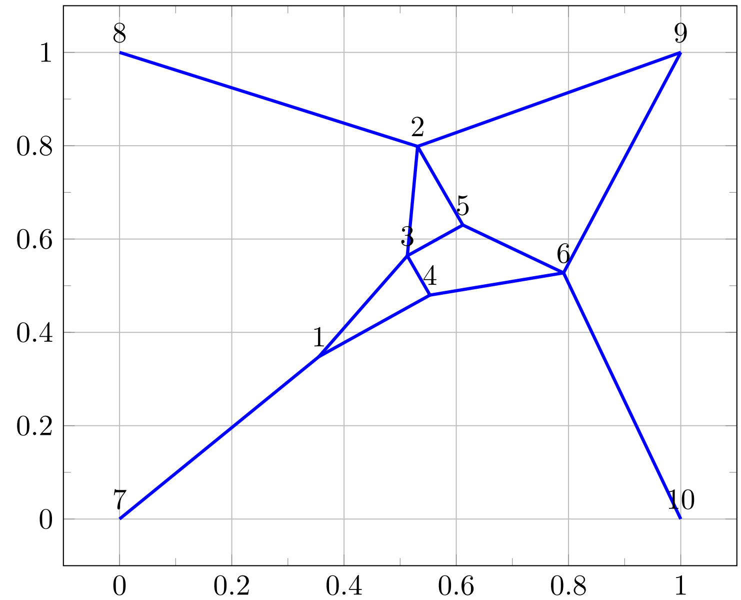
$$A = \left[\begin{array}{cc} B_{\mathrm{m}}^T & 0 \\ 0 & B_{\mathrm{m}}^T \end{array} \right], \qquad b = -\left[\begin{array}{c} B_{\mathrm{f}}^T u_{\mathrm{f}} \\ B_{\mathrm{f}}^T v_{\mathrm{f}} \end{array} \right].$$

   Note that the problem is equivalent to two independent least squares problems

$$\text{minimize} \quad \|B_{\mathrm{m}}^T u_{\mathrm{m}} + B_{\mathrm{f}}^T u_{\mathrm{f}}\|^2, \qquad \text{minimize} \quad \|B_{\mathrm{m}}^T v_{\mathrm{m}} + B_{\mathrm{f}}^T v_{\mathrm{f}}\|^2,$$

   with variables u_{m} and v_{m}, respectively, and that these two problems have the same coefficient matrix B_{m}^T.

(c) The solution is shown in the figure.



A similar figure is produced by the following MATLAB code.

```
E = [ 1, 3;  1, 4;  1, 7;  2, 3;  2, 5;  2, 8;  2, 9;
      3, 4;  3, 5;  4, 6;  5, 6;  6, 9;  6, 10 ];
m = size(E, 1);
n = 10;
B = zeros(n, m);
for k = 1:m
    B(E(k,1), k) = 1;
    B(E(k,2), k) = -1;
end;
fixed = 7:10;
pos_fixed = [0,0; 0,1; 1,1; 1,0];
free = 1:6;
pos_free = -B(free,:)' \ ( B(fixed,:)' * pos_fixed );
plot(pos_free(:,1), pos_free(:,2), 'o',
    pos_fixed(:,1), pos_fixed(:,2), 'ro');
hold on
pos = [pos_free; pos_fixed];
for k=1:n
    text(pos(k,1), pos(k,2), int2str(k));
end
for k = 1:m
    plot([pos(E(k,1), 1), pos(E(k,2), 1)], ...
         [pos(E(k,1), 2), pos(E(k,2), 2)], 'k-');
end;
```

2. *Exercise T13.3.*

(a) We minimize the sum of the squares of the prediction errors,

$$\sum_{k=1}^{13} (\log_{10} n_k - \theta_1 - (t_k - 1970)\theta_2)^2 = \|A\theta - b\|^2,$$

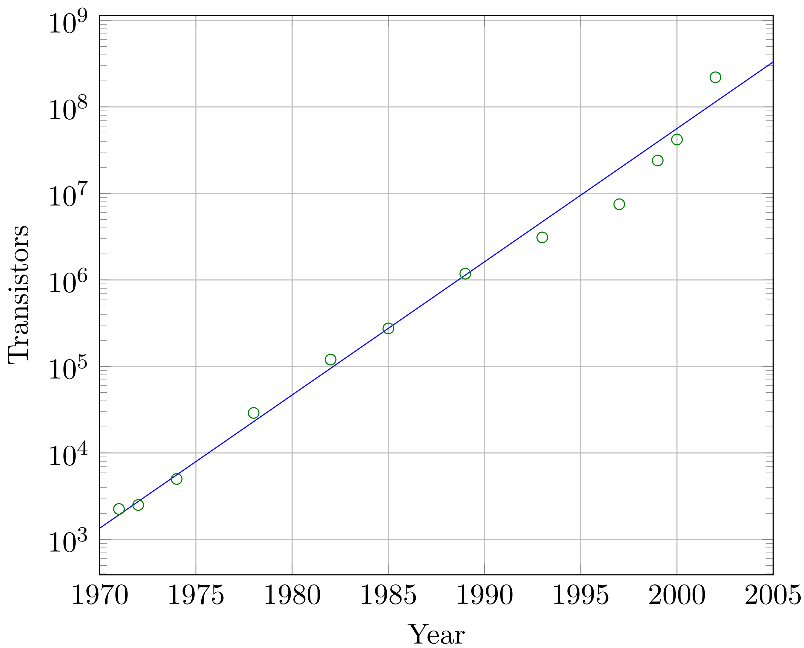
where

$$A = \begin{bmatrix} 1 & t_1 - 1970 \\ 1 & t_2 - 1970 \\ \vdots & \vdots \\ 1 & t_{13} - 1970 \end{bmatrix}, \qquad b = \begin{bmatrix} \log_{10} N_1 \\ \log_{10} N_2 \\ \vdots \\ \log_{10} N_{13} \end{bmatrix}.$$

The solution is $\hat{\theta}_1 = 3.13$ and $\hat{\theta}_2 = 0.154$. The RMS error of this model is

$$\frac{1}{\sqrt{13}} \|A\hat{\theta} - b\| = 0.20.$$

This means that we can expect our prediction of $\log_{10} N$ to typically be off by around 0.20. This corresponds to a prediction typically off by around a factor of $10^{0.2} = 1.6$. The straight-line fit is shown in the following figure.



(b) The predicted number of transistors in 2015 is

$$10^{\theta_1 + \theta_2(2015-1970)} \approx 1.14 \times 10^{10}$$

This prediction is about a factor of 3 off from the IBM Z13 processor. That is around two standard deviations off from the prediction, which is reasonable. (Although in general we would not expect extrapolations to have the same error as observed on the training data set.)

(c) In our model the number of transistors doubles approximately every $(\log_{10} 2)/\theta_2 = 1.95$ years, which is consistent with Moore's law.

3. *Exercise A8.3.* The inverse of the nonlinear function $h(x) = e^x/(1+e^x)$ is $h^{-1}(y) = \log(y/(1-y))$, *i.e.*,

$$y = \frac{e^x}{1+e^x} \qquad \Longleftrightarrow \qquad x = \log\left(\frac{y}{1-y}\right).$$

Applying this nonlinear transformation to the two sides of equation (15) in the assignment gives a linear set of equations

$$\alpha t_i + \beta \approx \log\left(\frac{y_i}{1-yi}\right), \qquad i = 1, \ldots, m.$$

This means that if we use as error function

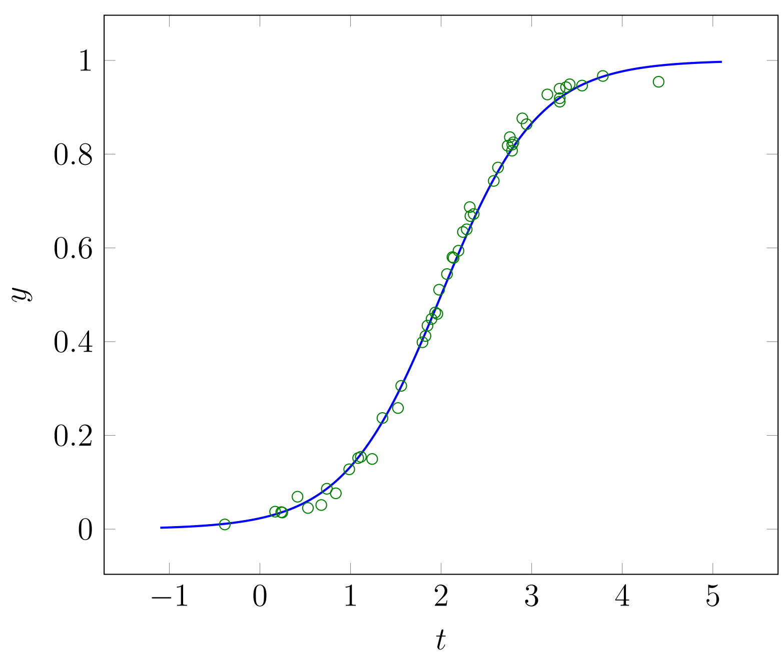
$$\sum_{i=1}^{m} \left(\alpha t_i + \beta - \log(\frac{y_i}{1-y_i})\right)^2$$

we get a least squares problem with

$$x = \begin{bmatrix} \alpha \\ \beta \end{bmatrix}, \qquad A = \begin{bmatrix} t_1 & 1 \\ t_2 & 1 \\ \vdots & \vdots \\ t_m & 1 \end{bmatrix}, \qquad b = \begin{bmatrix} \log(y_1/(1-y_1)) \\ \log(y_2/(1-y_2)) \\ \vdots \\ \log(y_m/(1-y_m)) \end{bmatrix}.$$

The solution on the data in `logisticfit.m` is

$$\alpha = 1.8676, \qquad \beta = -3.7397.$$



4. *Exercise A8.8.*

(a) We use the property that a square matrix B is nonsingular if and only if $Bx = 0$ implies $x = 0$.

Suppose $x \in \mathbf{R}^m$ and $y \in \mathbf{R}^n$ satisfy

$$\begin{bmatrix} I & A \\ A^T & 0 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = 0,$$

i.e., $x + Ay = 0$ and $A^T x = 0$. Then

$$A^T x = -A^T A y = 0.$$

The matrix $A^T A$ is nonsingular, because it is the Gram matrix of a matrix with linearly independent columns. Therefore $y = 0$ and $x = -Ay = 0$.

(b) The equations

$$\begin{bmatrix} I & A \\ A^T & 0 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} b \\ 0 \end{bmatrix}$$

are equivalent to

$$x + Ay = b, \qquad A^T x = 0.$$

We have to check that those two equations are satisfied for

$$x = b - Ax_{\rm ls}, \qquad y = x_{\rm ls}.$$

Clearly this choice of x and y satisfies $x + Ay = b$. Moreover, we have

$$\begin{aligned} A^T x &= A^T(b - A(A^T A)^{-1} A^T b) \\ &= A^T b - (A^T A)(A^T A)^{-1} A^T b \\ &= A^T b - A^T b \\ &= 0. \end{aligned}$$

5. *Exercise A8.12.*

(a) We verify that $\hat{y}$ satisfies the normal equations for the least squares problem

$$\text{minimize} \quad \left\| \begin{bmatrix} A \\ c^T \end{bmatrix} y - \begin{bmatrix} b \\ d \end{bmatrix} \right\|^2 = \|Ay - b\|^2 + (c^T y - d)^2.$$

The normal equations are

$$\begin{bmatrix} A \\ c^T \end{bmatrix}^T \begin{bmatrix} A \\ c^T \end{bmatrix} \hat{y} = \begin{bmatrix} A \\ c^T \end{bmatrix}^T \begin{bmatrix} b \\ d \end{bmatrix},$$

or

$$(A^T A + cc^T)\hat{y} = A^T b + dc.$$

We verify that the proposed $\hat{y}$ satisfies this equation:

$$
\begin{aligned}
(A^TA + cc^T)\hat{y} &= (A^TA + cc^T)\left(\hat{x} + \frac{d - c^T\hat{x}}{1 + c^T(A^TA)^{-1}c}(A^TA)^{-1}c\right) \\
&= A^Tb + (c^T\hat{x})c + \frac{d - c^T\hat{x}}{1 + c^T(A^TA)^{-1}c}(1 + c^T(A^TA)^{-1}c)c \\
&= A^Tb + (c^T\hat{x})c + (d - c^T\hat{x})c \\
&= A^Tb + dc.
\end{aligned}
$$

On the second line we used $A^TA\hat{x} = A^Tb$.

(b) If we plug in the QR factorization of A, the formulas for $\hat{x}$ and $\hat{y}$ reduce to

$$
\hat{x} = R^{-1}Q^Tb, \qquad \hat{y} = \hat{x} + \frac{d - c^T\hat{x}}{1 + c^TR^{-1}R^{-T}c}R^{-1}R^{-T}c.
$$

The two vectors can be computed as follows.

- QR factorization of A. $2mn^2$ flops.
- Matrix-vector product $u = Q^Tb$. $2mn$ flops.
- Compute $\hat{x} = R^{-1}u$ using back substitution. n^2 flops.
- Compute $v = R^{-T}c$ using forward substitution. n^2 flops.
- Compute $\alpha = (d - c^T\hat{x})/(1 + v^Tv)$. $4n$ flops.
- Compute $w = R^{-1}v$ using back substitution. n^2 flops.
- Compute $\hat{y} = \hat{x} + \alpha w$. $2n$ flops.

The total is $2mn^2 + 2mn + 3n^2 + 6n \approx 2mn^2$.

6. *Exercise A8.13.*

(a) At optimum $z = b_k - a_k^Ty$ and the kth term vanishes.

(b) The problem in part (a) can be written in matrix form as

$$
\text{minimize} \quad \left\| \begin{bmatrix} A & e_k \end{bmatrix} \begin{bmatrix} y \\ z \end{bmatrix} - b \right\|^2
$$

where e_k is the kth unit vector. The normal equations are

$$
\begin{bmatrix} A & e_k \end{bmatrix}^T \begin{bmatrix} A & e_k \end{bmatrix} \begin{bmatrix} y \\ z \end{bmatrix} = \begin{bmatrix} A & e_k \end{bmatrix}^T b.
$$

This simplifies to

$$
\begin{bmatrix} A^TA & a_k \\ a_k^T & 1 \end{bmatrix} \begin{bmatrix} y \\ z \end{bmatrix} = \begin{bmatrix} A^Tb \\ b_k \end{bmatrix}.
$$

From the first equation

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

Substituting this in the second equation and solving for z gives

$$z = \frac{b_k - a_k^T \hat{x}}{1 - a_k^T (A^T A)^{-1} a_k}.$$

Therefore

$$y = \hat{x} - \frac{b_k - a_k^T \hat{x}}{1 - a_k^T (A^T A)^{-1} a_k} (A^T A)^{-1} a_k.$$

(c) We substitute the factorization $A = QR$ in the formula for $\hat{y}_k$:

$$\hat{y}_k = \hat{x} - \frac{b_k - a_k^T \hat{x}}{1 - a_k^T R^{-1} R^{-T} a_k} R^{-1} R^{-T} a_k.$$

Note that $a_k^T R^{-1} = e_k^T A R^{-1} = e_k^T Q$ is the kth row q_k^T of Q, so

$$\hat{y}_k = \hat{x} - \frac{b_k - a_k^T \hat{x}}{1 - q_k^T q_k} R^{-1} q_k.$$

The dominant terms in the algorithm are the QR factorization and the calculation of the m vectors $R^{-1} q_k$.

- QR factorization $A = QR$. $2mn^2$ flops.
- Compute the matrix-vector product $Q^T b$ and compute $\hat{x} = R^{-1} Q^T b$ by back substitution. $2mn + n^2$ flops.
- Compute the vectors $v_k = R^{-1} q_k$ for $k = 1, \ldots, m$ by back substitution. $n^2 m$ flops.
- For $k = 1, \ldots, m$, compute $\alpha_k = (b_k - a_k \hat{x}_k)/(1 - q_k^T q_k)$. This requires a matrix vector product $A\hat{x}$ and m inner products of lenghth m. $4mn$ flops..
- For $k = 1, \ldots, m$, compute $\hat{y}_k = \hat{x} - \alpha_k v_k$. $2mn$ flops.

The total is $3mn^2$, *i.e.*, almost the same as just computing $\hat{x}$.