CS/ECE/ME 532
Homework 6: more SVD

due: Friday October 28, 2016

1. **Normal equations.** Using the SVD, show that for any $A \in \mathbb{R}^{m \times n}$ and $b \in \mathbb{R}^m$, the normal equations $A^\mathsf{T} A x = A^\mathsf{T} b$ always have at least one solution. What is this solution?

   SOLUTION: writing $A = U_1 \Sigma_1 V_1^\mathsf{T}$ (compact SVD), the normal equations are equivalent to:

   $$V_1 \Sigma_1^2 V_1^\mathsf{T} x = V_1 \Sigma_1 U_1^\mathsf{T} b$$

   The simplest way to show an equation always has a solution is to provide that solution and show it works. Since solutions to the normal equations are also solutions to the associated least squares problem and vice versa, we might try $x = A^\dagger b = V_1 \Sigma_1^{-1} U_1^\mathsf{T} b$, which we know to be a solution to the least squares problem. Substituting, we find:

   $$\begin{aligned} V_1 \Sigma_1^2 V_1^\mathsf{T} x &= \left(V_1 \Sigma_1^2 V_1^\mathsf{T} \right) \left(V_1 \Sigma_1^{-1} U_1^\mathsf{T} b \right) \\ &= V_1 \Sigma_1^2 \Sigma_1^{-1} U_1^\mathsf{T} b \\ &= V_1 \Sigma_1 U_1^\mathsf{T} b \end{aligned}$$

   which is what we set out to show. So the normal equations always have at least one solution. The vector $x = A^\dagger b = V_1 \Sigma_1^{-1} U_1^\mathsf{T} b$ always satisfies the normal equations.

2. **Right inverses.** A *right inverse* of $A \in \mathbb{R}^{m \times n}$ is a matrix $B \in \mathbb{R}^{n \times m}$ such that $AB = I$.

   a) Prove that if the rows of A are linearly independent, A has a right inverse.

   b) Prove that if A has a right inverse, the rows of A are linearly independent.

   c) Suppose A has linearly independent rows. Find a parameterization of all right inverses of A.

   SOLUTION:

   a) If the rows of A are linearly independent, then AA^T is invertible. One possible choice for a right inverse is $B = A^\dagger = A^\mathsf{T}(AA^\mathsf{T})^{-1}$. We can check:

   $$AB = AA^\mathsf{T}(AA^\mathsf{T})^{-1} = I$$

   b) Suppose B is a right inverse of A. In the equation $AB = I$, we have $I \in \mathbb{R}^{m \times m}$, and $\text{rank}(I) = m$. Now $\text{rank}(AB) \leq \min(m, n)$. So it must be the case that $m \leq n$, i.e. A is a wide matrix.

   First proof: If A didn't have linearly independent rows, we would have $\text{rank}(A) = r < m$. And consequently, $\text{rank}(AB) \leq r < m = \text{rank}(I)$. So it would be impossible to satisfy $AB = I$. We conclude that A must have linearly independent rows.

   Alternate proof: If the rows of A are linearly dependent, then there is some $y \neq 0$ such that $y^\mathsf{T} A = 0$ (linear combination of the rows is zero). Multiplying the equation $AB = I$ by y^T on the left, we obtain: $0 = y^\mathsf{T}$, a contradiction. Therefore the rows of A must be linearly independent.

c) We are being asked to find all B that satisfy the equation $AB = I$. If we let $B = \begin{bmatrix} b_1 & \dots & b_m \end{bmatrix}$, then the equation $AB = I$, taken column-by-column, is: $Ab_i = e_i$ for $i = 1, \dots, m$ (e_i is the vector with a 1 in the i^{th} position and 0's everywhere else). For each of these equations, the set of all solutions is of the form $b_i = A^\dagger e_i + V_2 w$, where $V_2 \in \mathbb{R}^{n \times (n-m)}$ is a matrix whose columns are are an orthonormal basis for null(A). Assembling into a matrix, the set of all solutions to $AB = I$ is:

$$B = A^\dagger + V_2 W \qquad \text{where } W \in \mathbb{R}^{(n-m) \times m} \text{ is any matrix}$$

3. Recovering a blurred signal. Many sensing and imaging systems produce signals that may be slightly distorted or blurred (e.g., an out-of-focus camera). In such situations, algorithms are needed to deblur the data to obtain a more accurate estimate of the true signal. Suppose our true signal is a vector $x \in \mathbb{R}^n$, and the blurring produces a new signal $b \in \mathbb{R}^n$ according to the model:

$$b_i = \frac{1}{k}\left(x_i + x_{i-1} + \cdots + x_{i-k+1}\right) + w_i \qquad \text{for } i = 1, \dots, n$$

In other words, each b_i is the average of the past k values of x_i, plus some extra noise w_i. Note: in the above formula, treat x_j as zero when $j < 1$. The goal is to estimate x using b and A.

a) We can write the above equations in the more compact form: $b = Ax + w$. Write code that generates the $A \in \mathbb{R}^{n \times n}$ matrix as a function of n and k.

SOLUTION: For $k = 3$ for example, the A matrix looks like:

$$A = \frac{1}{3}\begin{bmatrix} 1 & 0 & 0 & 0 & \dots & 0 \\ 1 & 1 & 0 & 0 & \dots & 0 \\ 1 & 1 & 1 & 0 & \dots & 0 \\ 0 & 1 & 1 & 1 & & \vdots \\ \vdots & \ddots & \ddots & \ddots & \ddots & 0 \\ 0 & \dots & 0 & 1 & 1 & 1 \end{bmatrix}$$

Many ways to generate this using code. The easiest way I found was to use the `toeplitz` command in Matlab:

```
A = 1/k * toeplitz([ones(k,1);zeros(n-k,1)],[1;zeros(n-1,1)]);
```

Another way, which is perhaps more intuitive, is to start with a zero matrix and add nonzero entries row by row:

```
A = zeros(n);
for i = 1:n
   A( i, max(i-k+1,1):i ) = 1/k;
end
```

b) Suppose the true x is given in the file `xsignal.csv`. Generate b by using $k = 30$. To generate w, make each w_i normally distributed with standard deviation σ. For example, you can do this in matlab via: `w=sigma*randn(n,1)`. Plot x and b using $\sigma = 0.01$ and $\sigma = 0.1$.

c) Reconstruct x in the three following ways, and for each one plot the true x and its reconstruction.

(i) Ordinary least squares

(ii) Truncated SVD; only keep the largest m singular values of A and try different values of m.

(iii) Regularized (Tikhonov) least squares; try different values of the parameter λ.

SOLUTION:

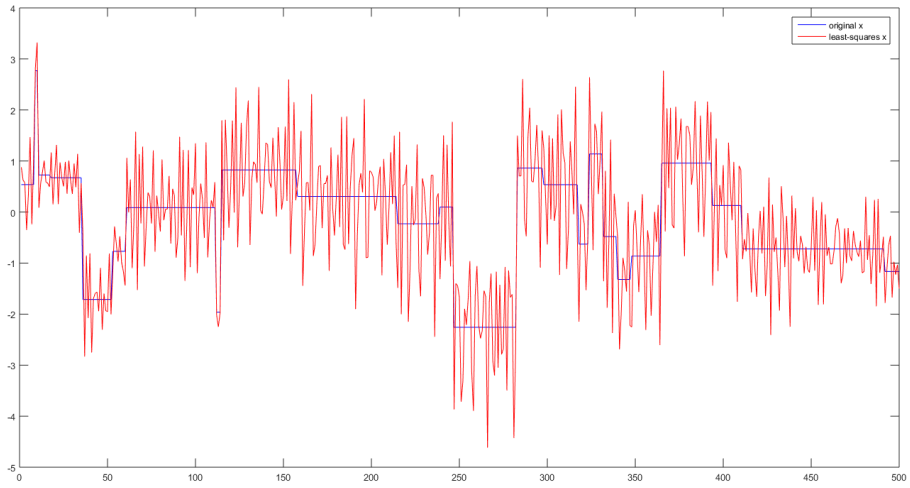
One easy way to implement each estimator is to start with A and b and compute the following for example:

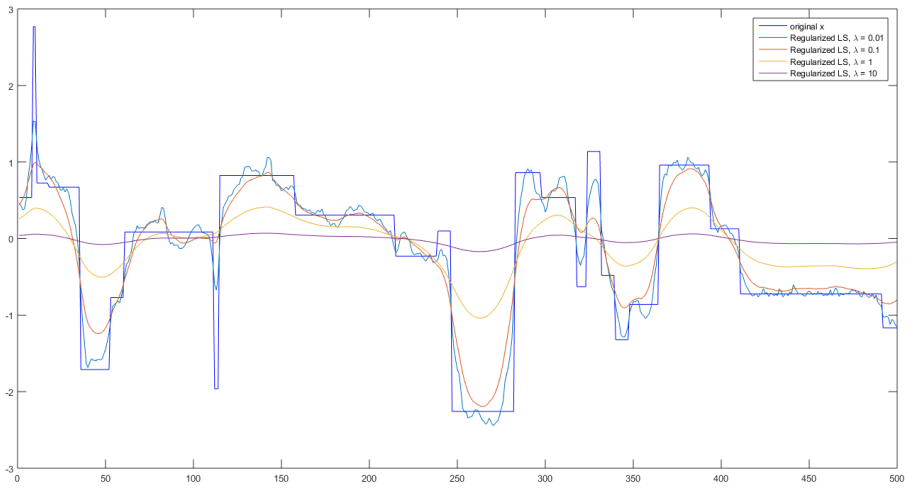
```
[U,S,V] = svd(A,'econ');

x_LS = V*diag( diag(S).^(-1) )*(U'*b);

% (loop over k values)
x_SVD = V(:,1:k)*diag( diag(S(1:k,1:k)).^(-1) )*(U(:,1:k)'*b);

% (loop over L (lambda) values)
x_RLS = V*diag( diag(S).*( (diag(S).^2+L).^(-1) ) )*(U'*b);
```

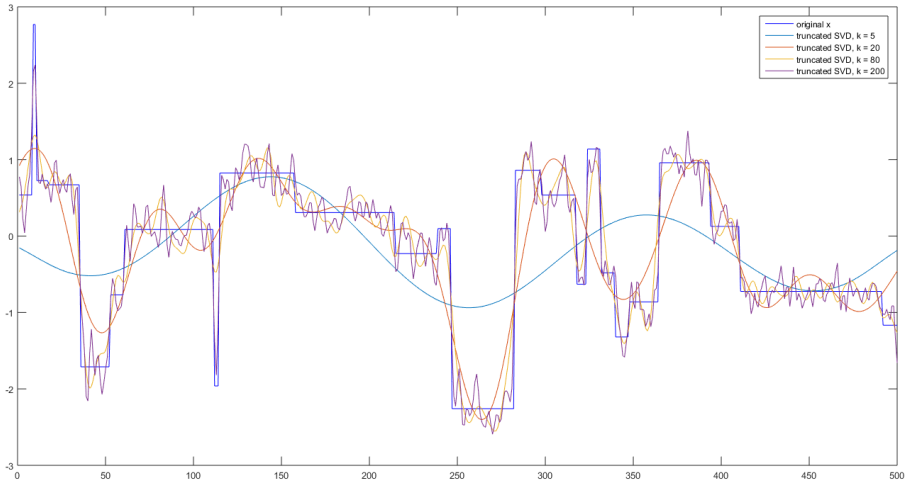
Note that I used `diag( diag(S).^(-1) )` rather than `inv(S)` since we know S is diagonal—it's more efficient to take the diagonal elements, invert them individually, and then rebuild a diagonal matrix.

d) Experiment with different averaging functions (i.e., different values of k in the code) and with different noise levels (σ in the code). How do the blurring and noise level affect the value of the regularization parameters that produce the best estimates?

SOLUTION: See the following page for sample estimators with default noise:







A great deal can be said about these estimators. Here are some examples of observations.

- The LS estimator is very noisy. This is because every singular value of A gets inverted. The smallest singular values are quite small, so when they get inverted they become large. These large values multiply the noisy $\boldsymbol{b}$ vector and ultimately amplify the noise. Even though the noise on $\boldsymbol{b}$ is relatively small, it gets amplified and this is evident in the estimate $\widehat{\boldsymbol{x}}$.
- The RLS estimator adds a regularization *before* inverting. So rather than computing $1/\sigma$ for each singular value of A, we compute $\sigma/(\sigma^2+\lambda)$, which is smaller than $1/\sigma$. This moves σ away from zero before inverting, which causes the noise to be amplified less. However, as λ gets larger the inverse is driven to zero. In the plot, we see that the limit $\lambda \to 0$ recovers the noisy LS estimate, and as λ gets larger the estimates are simultaneously de-noised and driven towards zero.
- The SVD estimator simply truncates the small singular values rather than inverting them. So as in the RLS case we are reducing the effect of noise, but without driving the estimate to zero. The result is akin to a Fourier series approximation, except the basis used is not the Fourier basis—it is a basis determined by the singular vectors. An important twist is that there is an additional trade-off: including more singular values increases the quality of the estimate, but also increases susceptibility to noise.