CS/ECE/ME 532
Homework 6: The SVD and Least Squares

1. Recall the face emotion classification problem from HW 3. Design and compare the performances of the classifiers proposed in **a** and **b**, below. In each case, divide the dataset into 8 equal sized subsets (e.g., examples $1-16$, $17-32$, etc). Use 6 sets of the data to estimate $\boldsymbol{x}$ for each choice of the *regularization parameter*, select the best value for the regularization parameter by estimating the error on one of the two remaining sets of data, and finally use the $\boldsymbol{x}$ corresponding to the best value of the regularization parameter to predict the labels of the remaining "hold-out" set. Compute the number of mistakes made on this hold-out set and divide that number by 16 (the size of the set) to estimate the error rate. Repeat this process 56 times (for the 8×7 different choices of the sets used to select the regularization parameter and estimate the error rate) and average the error rates to obtain a final estimate.

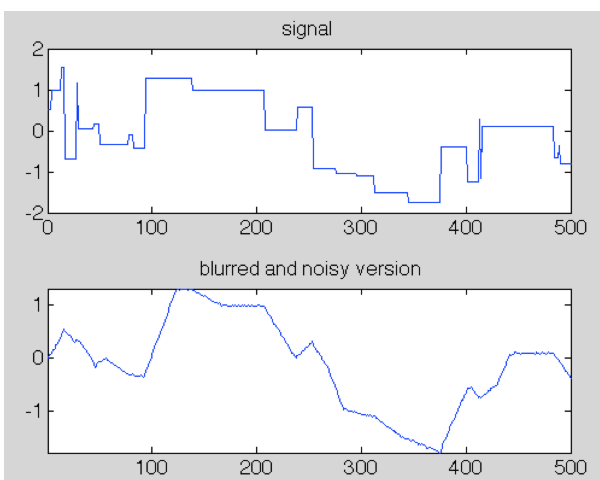
a. Truncated SVD solution. Use the pseudo-inverse $\boldsymbol{V}\boldsymbol{\Sigma}^{-1}\boldsymbol{U}^T$, where $\boldsymbol{\Sigma}^{-1}$ is computed by inverting the k largest singular values and setting all others to zero. In this case, k is the regularization parameter and it takes values $k = 1, 2, \ldots, 9$; i.e., compute 9 different solutions, $\widehat{\boldsymbol{x}}_k$, $k = 1, \ldots, 9$.

b. Regularized LS. Let $\widehat{\boldsymbol{x}}_\lambda = \arg\max_{\boldsymbol{x}} \|\boldsymbol{b} - \boldsymbol{A}\boldsymbol{x}\|_2^2 + \lambda\|\boldsymbol{x}\|_2^2$, for the following values of the regularization parameter $\lambda = 0, 2^{-1}, 2^0, 2^1, 2^2, 2^3$, and 2^4. Show that $\widehat{\boldsymbol{x}}_\lambda$ can be computed using the SVD and use this fact in your code.

2. 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. The Matlab code `blurring.m` generates a random signal and a blurred and noisy version of it, similar to the example shown below. The code simulates this equation:

$$\boldsymbol{b} = \boldsymbol{A}\boldsymbol{x} + \boldsymbol{e},$$

where $\boldsymbol{b}$ is the blurred and noisy signal, $\boldsymbol{A}$ is a matrix that performs the blurring operation, $\boldsymbol{x}$ is the true signal, and $\boldsymbol{e}$ is a vector of errors/noise. The goal is to estimate $\boldsymbol{x}$ using $\boldsymbol{b}$ and $\boldsymbol{A}$.



a. Implement the standard LS, truncated SVD, and regularized LS methods for this problem.

b. 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?

CS/ECE/ME 532

Homework 6 Solutions: The SVD and Least Squares

October 25, 2014

1 Problem1

(a) The Matlab code for Truncated SVD is given in the appendix below.

(b) Show that $\widehat{\boldsymbol{x}}_\lambda$ can be computed using the SVD and use this fact in your code.

Proof. The solution to

$$\widehat{\boldsymbol{x}}_\lambda = \underset{\boldsymbol{x}}{\operatorname{argmin}} \|\boldsymbol{b} - \boldsymbol{A}\boldsymbol{x}\|_2^2 + \lambda \|\boldsymbol{x}\|_2^2$$

is given by

$$\widehat{\boldsymbol{x}}_\lambda = \left(\boldsymbol{A}^\mathsf{T}\boldsymbol{A} + \lambda\boldsymbol{I}\right)^{-1}\boldsymbol{A}^\mathsf{T}\boldsymbol{b}. \tag{1}$$

If we let $\boldsymbol{A} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^\mathsf{T}$, then we may write

$$\boldsymbol{A}^\mathsf{T}\boldsymbol{A} = \left(\boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^\mathsf{T}\right)^\mathsf{T}\left(\boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^\mathsf{T}\right) = \boldsymbol{V}\boldsymbol{\Sigma}\underbrace{\boldsymbol{U}^\mathsf{T}\boldsymbol{U}}_{\boldsymbol{I}}\boldsymbol{\Sigma}\boldsymbol{V}^\mathsf{T} = \boldsymbol{V}\boldsymbol{\Sigma}^2\boldsymbol{V}^\mathsf{T}.$$

Plugging this in (1) we obtain

$$\begin{aligned}
\widehat{\boldsymbol{x}}_\lambda &= \left(\boldsymbol{V}\boldsymbol{\Sigma}^2\boldsymbol{V}^\mathsf{T} + \lambda\boldsymbol{V}\boldsymbol{I}\boldsymbol{V}^\mathsf{T}\right)^{-1}\cdot\left(\boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^\mathsf{T}\right)^\mathsf{T}\boldsymbol{b} \\
&= \boldsymbol{V}\left(\boldsymbol{\Sigma}^2 + \lambda\boldsymbol{I}\right)^{-1}\underbrace{\boldsymbol{V}^\mathsf{T}\boldsymbol{V}}_{\boldsymbol{I}}\boldsymbol{\Sigma}\boldsymbol{U}^\mathsf{T}\boldsymbol{b} \\
&= \boldsymbol{V}\left(\boldsymbol{\Sigma}^2 + \lambda\boldsymbol{I}\right)^{-1}\boldsymbol{\Sigma}\boldsymbol{U}^\mathsf{T}\boldsymbol{b}.
\end{aligned}$$

□

The Matlab code for Regularized LS can be found in the appendix below.

2 Problem 2

(a) The implementation of the standard LS, truncated SVD, and regularized LS methods can be found in the appendix below.

(b) Regularization is most effective under significant noise or blurring. There is an inverse relation between noise and the most convenient truncation.

Appendix

```
clear all;
load face_emotion_data

mistakes_tSVD = zeros(8,7);
mistakes_RLS = zeros(8,7);

X_tSVD = zeros(9,9);
X_RLS = zeros(9,7);

lambda = [0 .5 1 2 4 8 16];
for i=1:8, %Hold-out set
    %Hold-out set
    Ai_Hold = A((i-1)*16+1:i*16,:);
    bi_Hold = b((i-1)*16+1:i*16);

    %Remove Hold-out set
    Ai = A;
    Ai((i-1)*16+1:i*16,:) = [];

    bi = b;
    bi((i-1)*16+1:i*16) = [];

    for j=1:7,  %Parameter choice set

        %Set that will be used to pick the best parameter
        Aj = Ai((j-1)*16+1:j*16,:);
        bj = bi((j-1)*16+1:j*16,:);

        %Remove set that will be used to pick the best parameter
        %Aij and Bij are the 6 sets of data used to estimate x.
        Aij = Ai;
        Aij((j-1)*16+1:j*16,:) = [];

        bij = bi;
        bij((j-1)*16+1:j*16) = [];

        %Compute SVD
        [U,S,V] = svd(Aij);

        %Standard LS would simply be: X_SVD = V*pinv(S)*U'*bij;

        %Truncated SVD
        invS = inv(S(1:9,1:9));
```

```
        tS = zeros(96,9);
        for k=1:9,
            tS(1:k,1:k) = invS(1:k,1:k);%This is our truncated Sigma
            X_tSVD(:,k) = V*tS'*U'*bij;  %This is our k^th estimate of x
        end

        %Regularized LS
        for ell=1:7,
            %This is our lambda^th estimate of x
            X_RLS(:,ell) = V/(S'*S + lambda(ell)*eye(9))*S'*U'*bij;
        end

        %Use the j^th set of data to choose best parameter

        %These are our estimates of bj using each classifier
        bj_tSVD = sign(Aj*X_tSVD);
        bj_RLS = sign(Aj*X_RLS);

        %Number of errors of each parameter and each classifier
        err_tSVD = sum(abs(bj_tSVD~=repmat(bj,1,9)));
        err_RLS = sum(abs(bj_RLS~=repmat(bj,1,7)));

        %Choose best parameter in each case
        [mistakes_tSVDj, kj] = min(err_tSVD);
        [mistakes_RLSj, lambdaj] = min(err_RLS);

        %Use best parameter to estimate b on Hold-out set
        bi_tSVD = sign(Ai_Hold*X_tSVD(:,kj));
        bi_RLS = sign(Ai_Hold*X_RLS(:,lambdaj));

        %Number of mistakes on Hold-out set
        mistakes_tSVD(i,j) = sum(abs(bi_tSVD - bi_Hold));
        mistakes_RLS(i,j) = sum(abs(bi_RLS - bi_Hold));

    end
end

rate_tSVD = mistakes_tSVD./16;
overallErrorRate_tSVD = mean(mean(rate_tSVD))

rate_RLS = mistakes_RLS./16;
overallErrorRate_RLS = mean(mean(rate_RLS))
```