ECE 532 - HW5

FALL 2017

1 (b) $\hat{w}_\lambda = \underset{w}{\operatorname{argmin}} \; \| y - Xw \|_2^2 + \lambda \|w\|_2^2$

$$\Rightarrow \quad \hat{w}_\lambda = (X^T X + \lambda I)^{-1} \cdot X^T y$$

writing $X = U \Sigma V^T$, consider $X^T X$

$$X^T X = (U \Sigma V^T)^T (U \Sigma V^T)$$

$$= V \Sigma \underbrace{U^T U}_{= I} \Sigma V^T$$

$$= V \Sigma^2 V^T$$

$$\therefore \quad \hat{w}_\lambda = (V \Sigma^2 V^T + \lambda I)^{-1} \cdot (U \Sigma V^T)^T \cdot y$$

$$= (V \Sigma^2 V^T + \lambda V \cdot V^T)^{-1} \cdot \underbrace{(U \Sigma V^T)^T}_{= V \Sigma U^T} \cdot y$$

$$= V (\Sigma^2 + \lambda I)^{-1} \cdot \underbrace{V^T (V}_{= I} \Sigma U^T) \cdot y$$

$$= V (\Sigma^2 + \lambda I)^{-1} \cdot \Sigma U^T \cdot y$$

ECE 532 - HW5 - Fall 2017 - Rebecca Willett

Table of Contents

Homework Assignment 5

Completed by Roumen Guha

```
close all
clear all
load hw3_532_fall17_face_emotion_data.mat
```

Problem 1

(a)

Returns the number of errors in the set.

```
countErrors = @(x,y) sum((sign(x) ~= sign(y)));

% The columns of W_truncatedSVD contain the 9 vectors of weights
W_truncatedSVD = zeros(9, 9);
errors_truncatedSVD = zeros(8, 7);

% Generate holdout indices
holdout = @(x) ((x - 1)*16 + 1):(x*16);

% Computed 8 * 7 times
for k = 1:8
    % Decide on the hold-out set to predict the labels (and measure
 the error)
    holdout_augmented_curr = holdout(k);

    % Grab the corresponding test data (and remove the hold-out set)
    y_test = y;
    y_test(holdout_augmented_curr) = [];

    X_test = X;
```

```
    X_test(holdout_augmented_curr, :) = [];

    % Grab the corresponding hold-out data
    y_holdout = y(holdout_augmented_curr);
    X_holdout = X(holdout_augmented_curr, :);

    for l = 1:7
        % Decide on the hold-out set to be used to decide on the
        % regularization paramater
        holdout_augmented_curr2 = holdout(l);

        % Again, grab the test data
        y_test2 = y_test;
        y_test2(holdout_augmented_curr2) = [];

        X_test2 = X_test;
        X_test2(holdout_augmented_curr2, :) = [];

        % Again, grab the corresponding hold-out data
        y_holdout2 = y(holdout_augmented_curr2);
        X_holdout2 = X(holdout_augmented_curr2, :);

        % Compute the SVD of this set
        [U, S, V] = svd(X_test2);

        % Truncate the SVD. Invert the k-largest singular values and
set
        % the others to zero. k is the regularization paramater and it
        % takes values 1,2,...,9.
        S_inverted = inv(S(1:9, 1:9));
        S_inverted_truncated = zeros(96, 9);
        for m = 1:9
            % For each value k, pick the inverted k-largest singular
            % values, while the others are set to zero.
            S_inverted_truncated(1:m, 1:m) = S_inverted(1:m, 1:m);
            W_truncatedSVD(:, m) = (V * S_inverted_truncated' * U') *
y_test2;
        end

        % Compute and classify the estimates of y_holdout2
        y_holdout2_truncatedSVD = sign(X_holdout2 * W_truncatedSVD);

        % Calculate the error of the truncated SVD
        error_truncatedSVD = countErrors(y_holdout2_truncatedSVD,
repmat(y_holdout2, 1, 9));

        % Find index of best weight vector
        [minError_truncatedSVD, m] = min(error_truncatedSVD);

        % Compute and classify the predicted labels with our best
weight vector
        y_holdout_truncatedSVD = sign(X_holdout * W_truncatedSVD(:,
m));
```

```
        % Store the number of mistakes to calculate the error
        errors_truncatedSVD(k, l) =
 countErrors(y_holdout_truncatedSVD, y_holdout);
    end
end

% Caclulate the final estimate of the error rate
finalErrorRate_truncatedSVD = mean(mean(errors_truncatedSVD ./ 16))


finalErrorRate_truncatedSVD =

    0.1094
```

(b)

```
lambda = [0, 0.5, 1, 2, 4, 8, 16];

% The columns of W_truncatedSVD contain the 9 vectors of weights
W_regularizedLeastSquares = zeros(9, 7);
errors_regularizedLeastSquares = zeros(8, 7);

% Computed 8 * 7 times
for k = 1:8
    % Decide on the hold-out set to predict the labels (and measure
 the error)
    holdout_augmented_curr = holdout(k);

    % Grab the corresponding test data (and remove the hold-out set)
    y_test = y;
    y_test(holdout_augmented_curr) = [];

    X_test = X;
    X_test(holdout_augmented_curr, :) = [];

    % Grab the corresponding hold-out data
    y_holdout = y(holdout_augmented_curr);
    X_holdout = X(holdout_augmented_curr, :);

    for l = 1:7
        % Decide on the hold-out set to be used to decide on the
        % regularization paramater
        holdout_augmented_curr2 = holdout(l);

        % Again, grab the test data
        y_test2 = y_test;
        y_test2(holdout_augmented_curr2) = [];

        X_test2 = X_test;
        X_test2(holdout_augmented_curr2, :) = [];

        % Again, grab the corresponding hold-out data
```

```
        y_holdout2 = y(holdout_augmented_curr2);
        X_holdout2 = X(holdout_augmented_curr2, :);

        % Compute the SVD of this set
        [U, S, V] = svd(X_test2);

        % Perform regularized least-squares
        for m = 1:7
            W_regularizedLeastSquares(:, m) = (V / (S' * S + lambda(m)
 * eye(9)) * S' * U') * y_test2;
        end

        % Compute and classify the estimates of y_holdout2
        y_holdout2_regularizedLeastSquares = sign(X_holdout2 *
 W_regularizedLeastSquares);

        % Calculate the error of the regularized least-squares
        error_regularizedLeastSquares =
 countErrors(y_holdout2_regularizedLeastSquares, repmat(y_holdout2, 1,
 7));

        % Find index of best weight vector
        [minError_regularizedLeastSquares, m] =
 min(error_regularizedLeastSquares);

        % Compute and classify the predicted labels with our best
 weight vector
        y_holdout_regularizedLeastSquares = sign(X_holdout *
 W_regularizedLeastSquares(:,m));

        % Store the number of mistakes to calculate the error
        errors_regularizedLeastSquares(k, l) =
 countErrors(y_holdout_regularizedLeastSquares, y_holdout);
    end
end

% Caclulate the final estimate of the error rate
finalErrorRate_regularizedLeastSquares =
 mean(mean(errors_regularizedLeastSquares ./ 16))
```

```
finalErrorRate_regularizedLeastSquares =

    0.0458
```

(c)

I predicted that the three additional columns won't be useful for classification. However, looking at the results, we see that generally the augmented data matrix does in fact reduce our error. I think this is because the error is considered non-zero if an element of our predicted y isn't exactly either -1 or +1. This then affect our future output, so by adding more datapoints to our overall optimization, we weight the calculations to more "true to life".

(c) (i)

```
X_augmented = [X, X*randn(9,3)];

[numRows, numCols] = size(X_augmented);

% The columns of W_truncatedSVD contain the 9 vectors of weights
W_augmented_truncatedSVD = zeros(numCols, numCols);
errors_augmented_truncatedSVD = zeros(numCols - 1, numCols - 2);

% Computed 8*7 times
for k = 1:8
    % Decide on the hold-out set to predict the labels (and measure
 the error)
    holdout_augmented_curr = holdout(k);

    % Grab the corresponding test data (and remove the hold-out set)
    y_augmented_test = y;
    y_augmented_test(holdout_augmented_curr) = [];

    X_augmented_test = X_augmented;
    X_augmented_test(holdout_augmented_curr, :) = [];

    % Grab the corresponding hold-out data
    y_augmented_holdout = y(holdout_augmented_curr);
    X_augmented_holdout = X_augmented(holdout_augmented_curr, :);

    for l = 1:7
        % Decide on the hold-out set to be used to decide on the
        % regularization paramater
        holdout_augmented_curr2 = holdout(l);

        % Again, grab the test data
        y_augmented_test2 = y_augmented_test;
        y_augmented_test2(holdout_augmented_curr2) = [];

        X_augmented_test2 = X_augmented_test;
        X_augmented_test2(holdout_augmented_curr2, :) = [];

        % Again, grab the corresponding hold-out data
        y_augmented_holdout2 = y(holdout_augmented_curr2);
        X_augmented_holdout2 =
 X_augmented(holdout_augmented_curr2, :);

        % Compute the SVD of this set
        [U, S, V] = svd(X_augmented_test2);

        % Truncate the SVD. Invert the k-largest singular values and
 set
        % the others to zero. k is the regularization paramater and it
        % takes values 1,2,...,9.
        S_inverted = inv(S(1:numCols, 1:numCols));
        S_inverted_truncated = zeros(96, numCols);
```

```
        for m = 1:numCols
            % For each value k, pick the inverted k-largest singular
            % values, while the others are set to zero.
            S_inverted_truncated(1:m, 1:m) = S_inverted(1:m, 1:m);
            W_augmented_truncatedSVD(:, m) = (V *
 S_inverted_truncated' * U') * y_augmented_test2;
        end

        % Compute and classify the estimates of y_holdout2
        y_holdout2_truncatedSVD = sign(X_augmented_holdout2 *
 W_augmented_truncatedSVD);

        % Calculate the error of the truncated SVD
        error_truncatedSVD = countErrors(y_holdout2_truncatedSVD,
 repmat(y_augmented_holdout2, 1, numCols));

        % Find index of best weight vector
        [minError_truncatedSVD, m] = min(error_truncatedSVD);

        % Compute and classify the predicted labels with our best
 weight vector
        y_holdout_truncatedSVD = sign(X_augmented_holdout *
 W_augmented_truncatedSVD(:, m));

        % Store the number of mistakes to calculate the error
        errors_augmented_truncatedSVD(k, l) =
 countErrors(y_holdout_truncatedSVD, y_augmented_holdout);
    end
end

% Caclulate the final estimate of the error rate
finalErrorRate_augmented_truncatedSVD =
 mean(mean(errors_augmented_truncatedSVD ./ 16))
```

```
finalErrorRate_augmented_truncatedSVD =

    0.0443
```

(c) (ii)

The columns of W_regularizedLeastSquares contain the 9 vectors of weights

```
W_augmented_regularizedLeastSquares = zeros(numCols, numCols);
errors_augmented_regularizedLeastSquares = zeros(numCols - 1, numCols
 - 2);

% Computed 8*7 times
for k = 1:8
    % Decide on the hold-out set to predict the labels (and measure
 the error)
    holdout_augmented_curr = holdout(k);
```

```
    % Grab the corresponding test data (and remove the hold-out set)
    y_augmented_test = y;
    y_augmented_test(holdout_augmented_curr) = [];

    X_augmented_test = X_augmented;
    X_augmented_test(holdout_augmented_curr, :) = [];

    % Grab the corresponding hold-out data
    y_augmented_holdout = y(holdout_augmented_curr);
    X_augmented_holdout = X_augmented(holdout_augmented_curr, :);

    for l = 1:7
        % Decide on the hold-out set to be used to decide on the
        % regularization paramater
        holdout_augmented_curr2 = holdout(l);

        % Again, grab the test data
        y_augmented_test2 = y_augmented_test;
        y_augmented_test2(holdout_augmented_curr2) = [];

        X_augmented_test2 = X_augmented_test;
        X_augmented_test2(holdout_augmented_curr2, :) = [];

        % Again, grab the corresponding hold-out data
        y_augmented_holdout2 = y(holdout_augmented_curr2);
        X_augmented_holdout2 =
X_augmented(holdout_augmented_curr2, :);

        % Compute the SVD of this set
        [U, S, V] = svd(X_augmented_test2);

        % Perform regularized least-squares
        for m = 1:7
            W_augmented_regularizedLeastSquares(:, m) = (V / (S' * S +
lambda(m) * eye(numCols)) * S' * U') * y_augmented_test2;
        end

        % Compute and classify the estimates of y_holdout2
        y_holdout2_regularizedLeastSquares = sign(X_augmented_holdout2
* W_augmented_regularizedLeastSquares);

        % Calculate the error of the regularized least-squares
        error_regularizedLeastSquares =
countErrors(y_holdout2_regularizedLeastSquares,
repmat(y_augmented_holdout2, 1, numCols));

        % Find index of best weight vector
        [minError_regularizedLeastSquares, m] =
min(error_regularizedLeastSquares);

        % Compute and classify the predicted labels with our best
weight vector
        y_holdout_regularizedLeastSquares = sign(X_augmented_holdout *
W_augmented_regularizedLeastSquares(:,m));
```

```
        % Store the number of mistakes to calculate the error
        errors_augmented_regularizedLeastSquares(k, l) =
 countErrors(y_holdout_regularizedLeastSquares, y_augmented_holdout);
    end
end

% Caclulate the final estimate of the error rate
finalErrorRate_augmented_regularizedLeastSquares =
 mean(mean(errors_augmented_regularizedLeastSquares ./ 16))
```

```
finalErrorRate_augmented_regularizedLeastSquares =

    0.0313
```

Published with MATLAB® R2017a

ECE 532 - HW5 - Fall 2017 - Rebecca Willett

Table of Contents

Homework Assignment 5

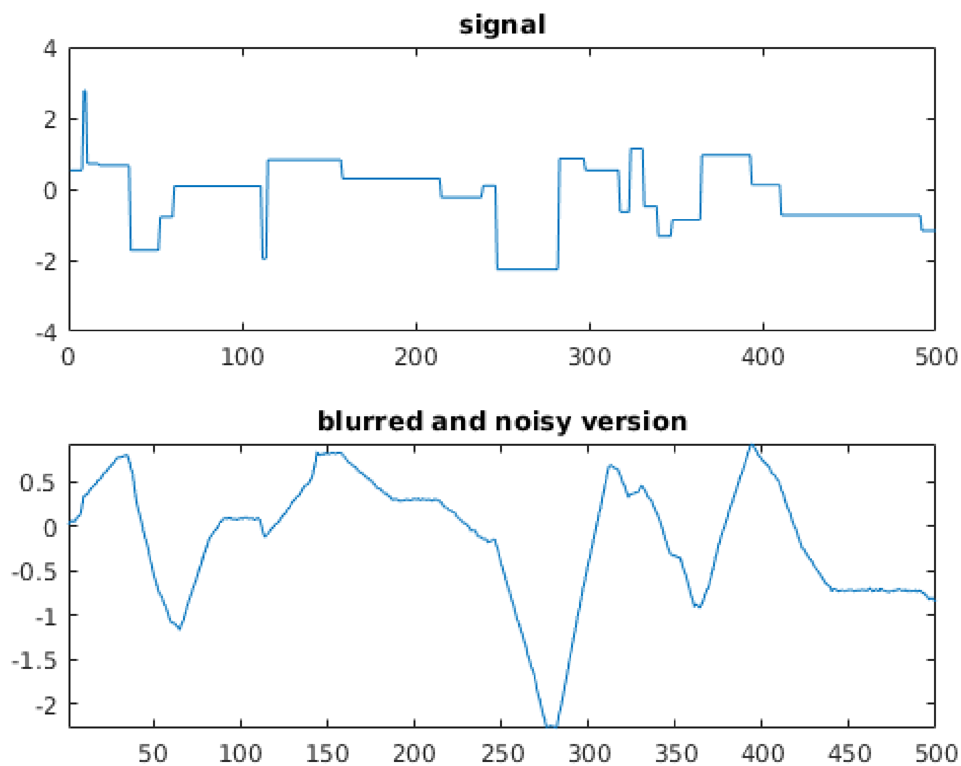
Completed by Roumen Guha

```
close all
clear all
```

Problem 2

```
rng(0) % initialize random seed

n = 500;
k = 30;
sigma = 0.01;

% generate random piecewise constant signal
w = zeros(n,1);
w(1) = randn;
for i = 2:n
    if (rand < 0.95)
        w(i) = w(i - 1);
    else
        w(i) = randn;
    end
end

% generate k-point averaging function
h = ones(1, k)/k;

% make X matrix for blurring
m = n+k-1;
for i = 1:m
    if i <= k
```

```
        X(i, 1:i) = h(1:i);
    else
        X(i, i-k+1:i) = h;
    end
end
X = X(:, 1:n);

% blurred signal + noise
y = X*w + sigma*randn(m, 1);

% plot
figure(1)
subplot(211)
plot(w)
title('signal')

subplot(212)
plot(y(1:n))
axis('tight')
title('blurred and noisy version')
```



(a)

```
errorRate = @(y, y_hat) sum((y - y_hat).^2);

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

(i) Standard Least-Squares

```
w_hat = (X' * X) \ X' * y;
y_hat = X * w_hat;

errorRate_leastSquares = errorRate(y, y_hat)
```

```
errorRate_leastSquares =

    0.0032
```

(ii) Truncated Singular Value Decomposition (SVD)

```
w_hat = (V * inv(S) * U') * y;
y_hat = X * w_hat;

errorRate_truncatedSVD = errorRate(y, y_hat)
```

```
errorRate_truncatedSVD =

    0.0032
```

(iii) Regularized Least-Squares (LS)

```
lambda = 0.0000001;

w_hat = (V / (S' * S + lambda * eye(500)) * S' * U') * y;
y_hat = X * w_hat;

errorRate_regularizedLS = errorRate(y, y_hat)
```

```
errorRate_regularizedLS =

    0.0032
```

(b)

It seems as though regularization works best when sigma and k are large, so when there is a lot of noise or blurring.

Published with MATLAB® R2017a