

E C E 5 3 2 - H W 5
F A L L 2 0 1 7

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

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

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

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

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

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

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

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

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

ECE 532 - HW5 - Fall 2017 - Rebecca Willett

Table of Contents

Homework Assignment 5	1
Problem 1	1
(a)	1
(b)	3
(c)	4
(c) (i)	5
(c) (ii)	6

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 parameter  
    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 parameter 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, 1) =
countErrors(y_holdout_truncatedSVD, y_holdout);
    end
end

% Caculate 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, 1) =
countErrors(y_holdout_regularizedLeastSquares, y_holdout);
end
end

% Caculate 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 affects 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, 1) =
countErrors(y_holdout_truncatedSVD, y_augmented_holdout);
end
end

% Caculate 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 parameter
    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, 1) =
countErrors(y_holdout_regularizedLeastSquares, y_augmented_holdout);
    end
end

% Caculate 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