CS/ECE/ME 532
Homework 7: regularization

due: Friday November 4, 2016

1. **Alternative regularization formulas.** This problem is about two alternative ways of solving the L_2-regularized least squares problem.

   a) Prove that for any $\lambda > 0$, the following matrix identity holds:

   $$(A^\mathsf{T} A + \lambda I)^{-1} A^\mathsf{T} = A^\mathsf{T}(AA^\mathsf{T} + \lambda I)^{-1}$$

   Hint: Start by considering the expression $A^\mathsf{T} AA^\mathsf{T} + \lambda A^\mathsf{T}$ and factor it in two different ways (from the right or from the left).

   b) The identity proved in part **a)** shows that there are actually two equivalent formulas for the solution to the L_2-regularized least squares problem. Generate random matrices $A \in \mathbb{R}^{8000\times 100}$ and $b \in \mathbb{R}^{8000}$, and find x that minimizes $\|Ax - b\|^2 + \lambda \|x\|^2$ in two different ways. Use $\lambda = 0.01$. Which formula computes more rapidly? Why? *Note:* you can use Matlab's `tic` and `toc` functions to measure execution time.

SOLUTION:

   a) Factoring the expression from the hint in two different ways, we have:

   $$A^\mathsf{T}(AA^\mathsf{T} + \lambda I) = (A^\mathsf{T} A + \lambda I)A^\mathsf{T}$$

   The bracketed terms are always invertible, so we can multiply on the right by $(AA^\mathsf{T} + \lambda I)^{-1}$ and on the left by $(A^\mathsf{T} A + \lambda I)^{-1}$ and we obtain the desired identity.

   b) The two equivalent formulas for the solution to the L_2-regularized least squares problem are:

   $$\widehat{x} = (A^\mathsf{T} A + \lambda I)^{-1} A^\mathsf{T} b = A^\mathsf{T}(AA^\mathsf{T} + \lambda I)^{-1} b$$

   The bottleneck in computation is the matrix inversion (or rather, finding the SVD). Since $A \in \mathbb{R}^{8000\times 100}$, this means $(A^\mathsf{T} A + \lambda I) \in \mathbb{R}^{100\times 100}$, whereas $(AA^\mathsf{T} + \lambda I) \in \mathbb{R}^{8000\times 8000}$. So the second formula will be much slower to compute.

2. **Getting a solver up and running.** Your task for this problem is to get a solver up and running and to solve a simple problem with it.

   - If you're using Matlab, install `CVX`: `http://cvxr.com/cvx/`
   - If you're using Python, install `CVXPY`: `http://www.cvxpy.org/en/latest/`
   - If you're using Julia, install `Convex`: `https://github.com/JuliaOpt/Convex.jl`

   Once your are up and running, write code to solve the regularized least squares problem:

   $$\underset{x,y}{\text{minimize}} \quad \left\| \begin{bmatrix} 1 & 0 \\ 2 & 1 \\ 1 & 3 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} - \begin{bmatrix} 4 \\ 5 \\ 7 \end{bmatrix} \right\|^2 + 0.1 \left\| \begin{bmatrix} x \\ y \end{bmatrix} \right\|^2$$

   Also solve this problem analytically and verify that you get the same answer.

SOLUTION: Here is an example in matlab:

```
%% problem data
A = [1 0; 2 1; 1 3];
b = [4; 5; 7];
lambda = 0.1;

%% solve using formula
xhat = (A'*A+lambda*eye(2))\(A'*b)

% OUTPUT:
% xhat =
%     2.2426
%     1.4641


%% solve using CVX
cvx_begin quiet
   variable x(2)
   minimize sum_square(A*x-b) + lambda*sum_square(x)
cvx_end
x

% OUTPUT:
% x =
%     2.2426
%     1.4641
```

3. Sparse classification. For this problem, we'll perform a more thorough analysis using the breast cancer dataset shown in class, `bcdata.csv`. The goal is to solve the Lasso problem

$$\underset{x \in \mathbb{R}^n}{\text{minimize}} \quad \|Ax - b\|^2 + \lambda \|x\|_1$$

Here x is the weight vector applied to the expression levels of $n = 8141$ genes and b is the vector of labels ($+1$ and -1) for each of the 295 patients. In the `bcdata.csv`, the first column corresponds to b and the rest of the columns are the A matrix. In this problem we will vary λ to explore the trade-off between data-fitting and sparsity, and we will compare the Lasso to Ridge Regression.

a) For each λ, find the optimal weights using only the first 100 patients (first 100 rows). Plot a trade-off curve with the squared residual $\|Ax - b\|^2$ on the vertical-axis and $\|x\|_1$ on the horizontal-axis. Your plot should contain the residual from the training data. Explain what you see. *Note:* you'll have to play around with how you choose λ to see the whole trade-off curve. When using all the columns of A, your code may take up to 30 seconds to run for each λ. So make sure your code is working properly (test it on a smaller subset of the columns first!) before you run it on the full dataset.

b) With your solutions from part **a)**, produce a new trade-off curve. This time, plot the error rate on the vertical-axis versus the sparsity on the horizontal-axis. Here, the error rate is the number of incorrect predictions divided by the total number of predictions. The sparsity is the number of nonzero entries in x. For this purpose, we'll say an entry x_i is nonzero if $|x_i| > 10^{-5}$. Again, the vertical-axis should contain the error rate from the training data.

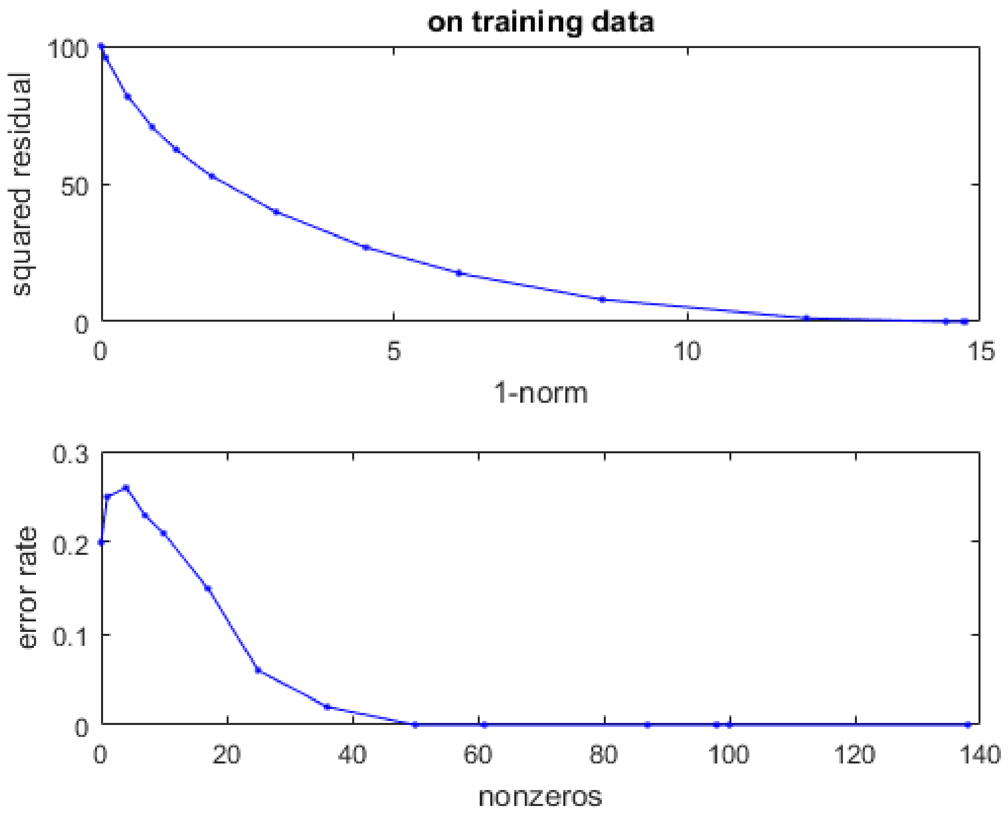
c) Repeat parts **a)** and **b)**. This time for the vertical-axis use the residual (respectively the error rate) from the validation data. For validation, use the remaining rows of the data matrix (the ones not used for training). Again, explain what you see in both trade-off plots.

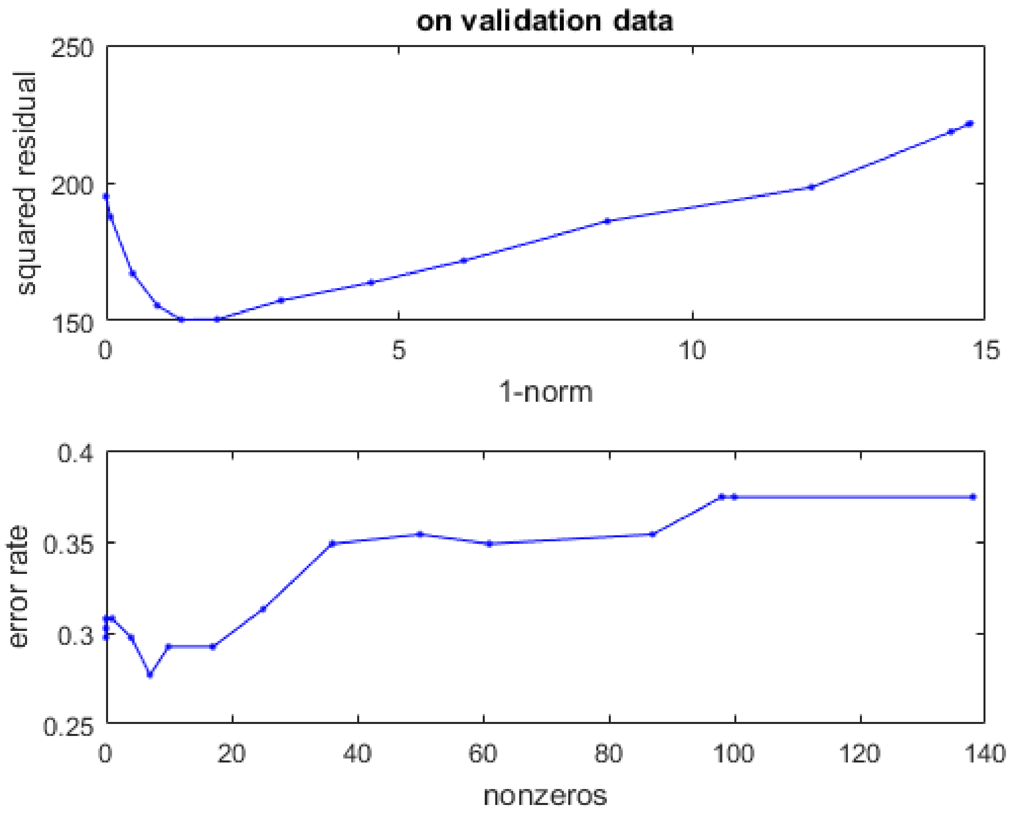
SOLUTION: The following Matlab code solves parts **a)** – **c)**.

```
% define training and validation sets
data = csvread('bcdata.csv');
A  = data(:,2:end);  b  = data(:,1);
At = A(1:100,:);     bt = b(1:100);
Av = A(101:end,:);   bv = b(101:end);
[m,n] = size(At);

% representative lambda values
lam_vals = [1e-6 1e-2 1e-1 1 3 5 7 10 14 18 23 30 50 70 80 90];
% lam_vals = [1e-2 1 5 10 30];
N = numel(lam_vals);

err = zeros(N,1); res = zeros(N,1); nrm = zeros(N,1);
nonz = zeros(N,1); errv = zeros(N,1); resv = zeros(N,1);

for i = 1:N
disp(i)
lambda = lam_vals(i);
cvx_begin quiet
variable x(n)
minimize sum_square(At*x-bt) + lambda*norm(x,1)
cvx_end
%x = At'*((At*At' + lambda*eye(m))\bt);
bhat = sign(At*x);
err(i) = mean( bhat ~= bt );
res(i) = norm(At*x-bt)^2;      % squared residual
nrm(i) = norm(x,1);            % size of 1-norm
nonz(i) = sum(abs(x) > 1e-5);  % number of nonzeros

bhatv = sign(Av*x);
errv(i) = mean( bhatv ~= bv );
resv(i) = norm(Av*x-bv)^2;
end

%% Figures
figure(1)
subplot(211); plot( nrm, res, 'b.-')
xlabel('1-norm'); ylabel('squared residual'); title('on training data')
subplot(212); plot( nonz, err, 'b.-')
xlabel('nonzeros'); ylabel('error rate')

figure(2)
subplot(211); plot( nrm, resv, 'b.-')
xlabel('1-norm'); ylabel('squared residual'); title('on validation data')
subplot(212); plot( nonz, errv, 'b.-')
xlabel('nonzeros'); ylabel('error rate')
```

And here are the plots. On the training set, the residual vs 1-norm looks convex as expected. And we see that the more we penalize the 1-norm, the more we encourage sparsity.

On validation, we see the effect of properly choosing λ. Also, something curious happens; there is a sweet spot where the estimator achieves both a good error rate as well as a sparse solution! The minimum error rate occurs with 7 nonzeros, which is *very* sparse considering we had over 8000 features. Of course, we only examined a limited number of λ values and it is possible we could find better solutions if we looked more thoroughly.


on training data
squared residual
100
50
0
0
5
10
15
1-norm
error rate
0.3
0.2
0.1
0
0
20
40
60
80
100
120
140
nonzeros


on validation data
squared residual
250
200
150
0
5
10
15
1-norm
error rate
0.4
0.35
0.3
0.25
0
20
40
60
80
100
120
140
nonzeros

d) Compare the performance of the Lasso and Ridge Regression in this application. Do this via the following steps.

(i) split the 295 patients into 10 subsets of size 29-30. We will use these subsets for training, tuning, and testing.

(ii) pick 8 of the subsets (about 240 patients) for training. Compute the solution $\widehat{x}$ for both the Lasso and Ridge Regression problems, each for a variety of λ values.

(iii) Out of the two remaining subsets, pick one for tuning. Evaluate your classifiers $\widehat{x}$ on the tuning data, and select the classifier with the smallest prediction error (one for Lasso and one for Ridge Regression).

(iv) finally, evaluate your best Lasso and Ridge Regression classifiers on the remaining subset of patients. How do they compare?

To get a better sense of the performance, you can repeat this entire process by taking different subsets of 8, 1, and 1 for training, tuning, and testing and compute the average squared errors and average number of misclassifications.

SOLUTION: Implementation for this problem is straightforward; we can swap out the CVX solver for the analytic solution to the ridge regression problem, or we can use CVX again, though it will be much slower. Note, as discussed in Problem 1, it is *much* more efficient to compute:

$$(A^\top A + \lambda I)^{-1} A^\top b = A^\top (AA^\top + \lambda I)^{-1} b$$

since the expression on the right requires inverting a 100×100 matrix while the one on the left requires inverting a 8141×8141 matrix. We can further accelerate the task by pre-computing the SVD of A.

We omit the plots in this solution, but for this problem, we find that ridge regression gives slightly worse performance in terms of error rate (but not substantially worse than using the Lasso). However ridge regression produces dense solutions while Lasso produces sparse ones.