CS/ECE/ME 532
Homework 6: Iterative Algorithms for Regularized LS

1. **Landweber convergence**. Consider the Landweber iteration for solving a standard least-squares problem with $\boldsymbol{A} \in \mathbb{R}^{m\times n}$, $\boldsymbol{b} \in \mathbb{R}^m$, and $\boldsymbol{A}$ has full column rank. Recall that this iteration begins with some initial $\boldsymbol{x}_0$ and then:

$$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k - \tau \boldsymbol{A}^\mathsf{T}(\boldsymbol{A}\boldsymbol{x}_k - \boldsymbol{b}) \qquad \text{for } k = 0, 1, \dots \tag{1}$$

a) We expect the algorithm to converge to $\boldsymbol{x}_\star = (\boldsymbol{A}^\mathsf{T}\boldsymbol{A})^{-1}\boldsymbol{A}^\mathsf{T}\boldsymbol{b}$. Define the *error* as $\boldsymbol{e}_k := \boldsymbol{x}_k - \boldsymbol{x}_\star$. Show how to rewrite (1) in the form $\boldsymbol{e}_{k+1} = \boldsymbol{P}\boldsymbol{e}_k$. What is the matrix $\boldsymbol{P}$?

SOLUTION: Substitute $\boldsymbol{x}_k \mapsto \boldsymbol{e}_k + \boldsymbol{x}_\star$ and $\boldsymbol{x}_{k+1} \mapsto \boldsymbol{e}_{k+1} + \boldsymbol{x}_\star$ and obtain:

$$\boldsymbol{e}_{k+1} = \boldsymbol{e}_k - \tau \boldsymbol{A}^\mathsf{T}(\boldsymbol{A}\boldsymbol{e}_k + \boldsymbol{A}\boldsymbol{x}_\star - \boldsymbol{b})$$

Then, using the fact that $\boldsymbol{A}^\mathsf{T}\boldsymbol{A}\boldsymbol{x}_\star = \boldsymbol{A}^\mathsf{T}\boldsymbol{b}$, the expression simplifies to:

$$\boldsymbol{e}_{k+1} = (\boldsymbol{I} - \tau\boldsymbol{A}^\mathsf{T}\boldsymbol{A})\boldsymbol{e}_k$$

Therefore $\boldsymbol{P} = \boldsymbol{I} - \tau\boldsymbol{A}^\mathsf{T}\boldsymbol{A}$.

b) Define the *residual* $\boldsymbol{r}_k := \boldsymbol{A}\boldsymbol{x}_k - \boldsymbol{b}$. Show how to rewrite (1) in the form $\boldsymbol{r}_{k+1} = \boldsymbol{Q}\boldsymbol{r}_k$. What is the matrix $\boldsymbol{Q}$?

SOLUTION: Compute the residual at time $k+1$ and obtain:

$$\begin{aligned}
\boldsymbol{r}_{k+1} &= \boldsymbol{A}\boldsymbol{x}_{k+1} - \boldsymbol{b}\\
&= \boldsymbol{A}(\boldsymbol{x}_k - \tau\boldsymbol{A}^\mathsf{T}(\boldsymbol{A}\boldsymbol{x}_k - \boldsymbol{b})) - \boldsymbol{b}\\
&= (\boldsymbol{I} - \tau\boldsymbol{A}\boldsymbol{A}^\mathsf{T})(\boldsymbol{A}\boldsymbol{x}_k - \boldsymbol{b})\\
&= (\boldsymbol{I} - \tau\boldsymbol{A}\boldsymbol{A}^\mathsf{T})\boldsymbol{r}_k
\end{aligned}$$

Therefore $\boldsymbol{Q} = \boldsymbol{I} - \tau\boldsymbol{A}\boldsymbol{A}^\mathsf{T}$.

c) Let $\{\sigma_i\}$ be the singular values of $\boldsymbol{A}$. Prove that when $0 < \tau < \frac{2}{\sigma_1^2}$, we have $\lim_{k\to\infty}\boldsymbol{e}_k = \boldsymbol{0}$. **Hint:** substitute the SVD of $\boldsymbol{A}$ into your expression for $\boldsymbol{P}$.

SOLUTION: Let $\boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^\mathsf{T} = \boldsymbol{A}$ be the full SVD of $\boldsymbol{A}$. The expression for $\boldsymbol{e}_{k+1}$ becomes:

$$\boldsymbol{e}_{k+1} = (\boldsymbol{I} - \tau\boldsymbol{V}\boldsymbol{\Sigma}^\mathsf{T}\boldsymbol{\Sigma}\boldsymbol{V}^\mathsf{T})\boldsymbol{e}_k$$

Multiply by $\boldsymbol{V}^\mathsf{T}$ on both sides and define $\boldsymbol{V}^\mathsf{T}\boldsymbol{e}_k = \boldsymbol{q}_k$ and rewrite as:

$$\boldsymbol{q}_{k+1} = (\boldsymbol{I} - \tau\boldsymbol{\Sigma}^\mathsf{T}\boldsymbol{\Sigma})\boldsymbol{q}_k$$

Since $\boldsymbol{V}^\mathsf{T}$ is orthogonal, $\boldsymbol{e}_k \to \boldsymbol{0}$ if and only if $\boldsymbol{q}_k \to \boldsymbol{0}$. And this, in turn, is equivalent to each component of $\boldsymbol{q}_k$ going to zero separately (because $\boldsymbol{I} - \tau\boldsymbol{\Sigma}^\mathsf{T}\boldsymbol{\Sigma}$ is diagonal). Moreover, $\boldsymbol{A}$ has full column rank, so $\boldsymbol{\Sigma}^\mathsf{T}\boldsymbol{\Sigma}$ is invertible—it is diagonal with entries $\sigma_1 \ge \cdots \ge \sigma_n > 0$. The equations corresponding to individual components of $\boldsymbol{q}_k$ look like: $q_{k+1} = (1 - \tau\sigma^2)q_k$. We have convergence to zero if and only if $|1 - \tau\sigma_i^2| < 1$ for all i. Rearranging, this is equivalent to: $0 < \tau < \frac{2}{\sigma_i^2}$. Since this must hold for all i, then because $\sigma_1 \ge \cdots \ge \sigma_n$, it must be the case that $0 < \tau < \frac{2}{\sigma_1^2}$.

d) Prove that if $\boldsymbol{A}$ is rank-deficient and $\boldsymbol{x}_0 = \boldsymbol{0}$, then the Landweber iteration converges to the minimum norm solution. **Hint:** redo part **a)** using $\boldsymbol{x}_\star = \boldsymbol{A}^\dagger \boldsymbol{b}$ and see how this affects part **c)**.

SOLUTION: The minimum-norm solution is $\boldsymbol{x}_\star = \boldsymbol{A}^\dagger \boldsymbol{b}$, which also satisfies $\boldsymbol{A}^\mathsf{T}\boldsymbol{A}\boldsymbol{x}_\star = \boldsymbol{A}^\mathsf{T}\boldsymbol{b}$. So the proof from part **a)** still holds here. Repeating the arguments from the proof of **c)**, we conclude as before that

$$\boldsymbol{q}_{k+1} = (\boldsymbol{I} - \tau \boldsymbol{\Sigma}^\mathsf{T}\boldsymbol{\Sigma})\boldsymbol{q}_k$$

The only difference is that this time, $\boldsymbol{A}$ is rank-deficient. Therefore, $\boldsymbol{\Sigma}^\mathsf{T}\boldsymbol{\Sigma}$ will have diagonal components $\sigma_1, \dots, \sigma_r, 0, \dots, 0$. For the first r components of $\boldsymbol{q}_k$, the same result holds as before; we have convergence to zero if and only if $0 < \tau < \frac{2}{\sigma_1^2}$. For the other components, we have: $q_{k+1} = q_k = \cdots = q_0$. Therefore, the only way these remaining components can go to zero is if they are always zero. Taking a closer look at the last $n - r$ components of $\boldsymbol{q}_0$,

$$\begin{aligned}
\begin{bmatrix} \boldsymbol{0} & \boldsymbol{I} \end{bmatrix} \boldsymbol{q}_0 &= \begin{bmatrix} \boldsymbol{0} & \boldsymbol{I} \end{bmatrix} \boldsymbol{V}^\mathsf{T}\left(\boldsymbol{x}_0 - \boldsymbol{A}^\dagger \boldsymbol{b}\right) \\
&= \begin{bmatrix} \boldsymbol{0} & \boldsymbol{I} \end{bmatrix} \boldsymbol{V}^\mathsf{T}\left(\boldsymbol{x}_0 - \boldsymbol{V}_1\boldsymbol{\Sigma}_1^{-1}\boldsymbol{U}_1^\mathsf{T}\boldsymbol{b}\right) \\
&= \boldsymbol{V}_2^\mathsf{T}\left(\boldsymbol{x}_0 - \boldsymbol{V}_1\boldsymbol{\Sigma}_1^{-1}\boldsymbol{U}_1^\mathsf{T}\boldsymbol{b}\right) \\
&= \boldsymbol{V}_2^\mathsf{T}\boldsymbol{x}_0
\end{aligned}$$

where we used in the last step that $\boldsymbol{V}_2^\mathsf{T}\boldsymbol{V}_1 = \boldsymbol{0}$. Since $\boldsymbol{x}_0 = 0$ by assumption, we have that $\begin{bmatrix} \boldsymbol{0} & \boldsymbol{I} \end{bmatrix} \boldsymbol{q}_0 = \boldsymbol{0}$. In other words, the components of $\boldsymbol{q}_k$ that don't shrink (τ has no effect on them) are zero anyway! This shows that $\boldsymbol{x}_k \to \boldsymbol{A}^\dagger\boldsymbol{b}$, as required. Note that $\boldsymbol{x}_0 = \boldsymbol{0}$ is a stronger condition than what's actually required for the convergence result. We will have $\boldsymbol{x}_k \to \boldsymbol{A}^\dagger\boldsymbol{b}$ if and only if $\boldsymbol{x}_0 = \boldsymbol{V}_1\boldsymbol{w}$ for some $\boldsymbol{w}$. Put another way, the result will be true if and only if $\boldsymbol{x}_0 \in \text{null}(\boldsymbol{A})^\perp$. In particular, it's true when $\boldsymbol{x}_0 = \boldsymbol{0}$.

2. Data Fitting vs. Sparsity Tradeoff. For this problem, we'll use the dataset `BreastCancer.mat` (see the journal paper posted on Moodle for more details about the dataset). The goal is to solve the Lasso problem

$$\underset{\boldsymbol{\beta} \in \mathbb{R}^n}{\text{minimize}} \quad \|\boldsymbol{X}\boldsymbol{\beta} - \boldsymbol{y}\|_2^2 + \lambda \|\boldsymbol{\beta}\|_1$$

Here $\boldsymbol{\beta}$ is the weight vector applied to the expression levels of 8141 genes and $\boldsymbol{y}$ is the vector of labels ($+1$ and -1) for each of the 295 patients. In this problem we will vary λ to explore the tradeoff between data-fitting and sparsity, and we will compare the Lasso to Ridge Regression.

a) Write an implementation of the ISTA (iterative soft-thresholding) as covered in class (and in the posted notes on the Moodle) that solves the above Lasso problem.

SOLUTION: See below for a simple Matlab implementation of iterative soft-thresholding.

```
function x = ista_solve( A, b, lambda )
%ista_solve: Iterative soft-thresholding!
%   this function solves the minimization problem
%   Minimize |Ax-b|_2^2 + lambda*|x|_1   (Lasso regression)
%   using iterative soft-thresholding.

   MAX_ITER = 1e6;          % maximum number of iterations
   TOL = 1e-5;              % convergence tolerance

   tau = 1/norm(A)^2;       % choose stepsize

   [~,n] = size(A);
   x = zeros(n,1);          % start point for the iteration

   for i = 1:MAX_ITER
      z = x - tau*(A'*(A*x-b));                            % Landweber
      xold = x;                                            % store old x
      x = sign(z) .* max( abs(z) - tau*lambda/2, 0 );      % ISTA
      if norm(x-xold) < TOL
         break
      end
   end
end
```

b) For each λ, use your code to find the optimal weights using only the first 100 patients (first 100 rows). Plot a trade-off curve with the residual $\|\boldsymbol{X}\boldsymbol{\beta} - \boldsymbol{y}\|_2$ on the vertical-axis and $\|\boldsymbol{\beta}\|_1$ on the horizontal-axis and. The vertical-axis 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 curve!

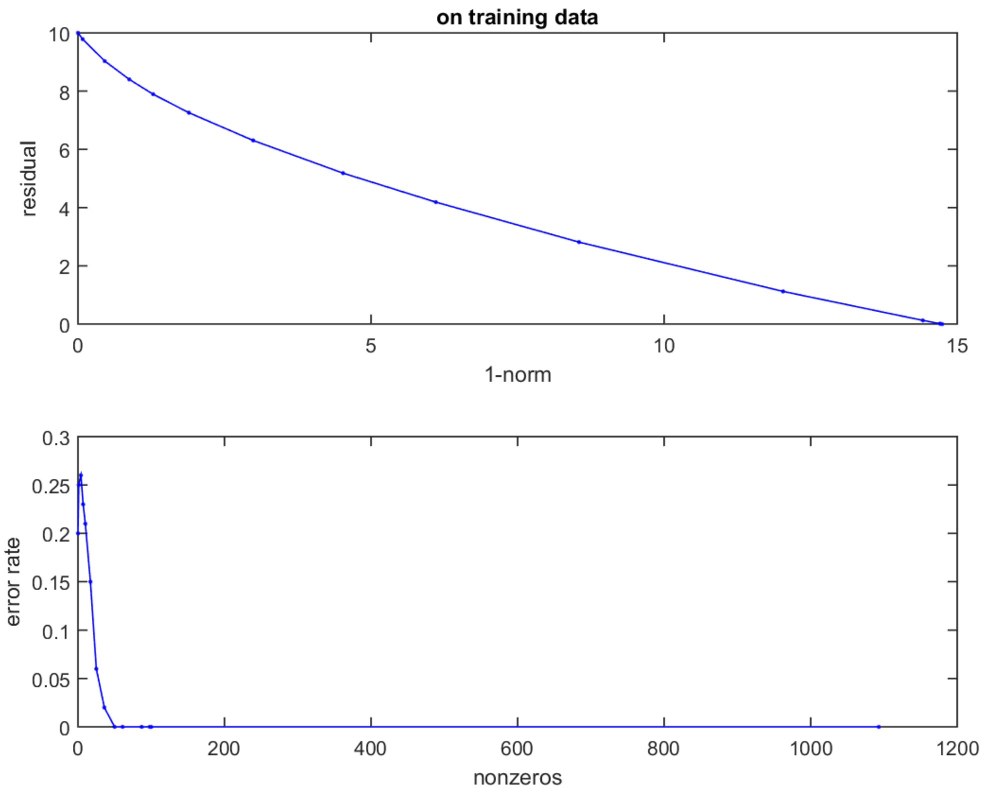
c) With your solutions from part **b)**, produce a new trade-off curve. This time, plot the error rate on the vertical-axis versus the sparsity on the horizontal-axis. Here, the error rate is the number of incorrect predictions divided by the total number of predictions. The sparsity is the number of nonzero entries in $\boldsymbol{\beta}$. For this purpose, we'll say an entry β_i is nonzero if $|\beta_i| > 10^{-6}$. Again, the vertical-axis should contain the error rate from the training data.

SOLUTION: The following Matlab code solves parts **b)** and **c)**.

```
load BreastCancer

At = X(1:100,:);  Av = X(101:end,:);
bt = y(1:100);  bv = y(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];
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);
```

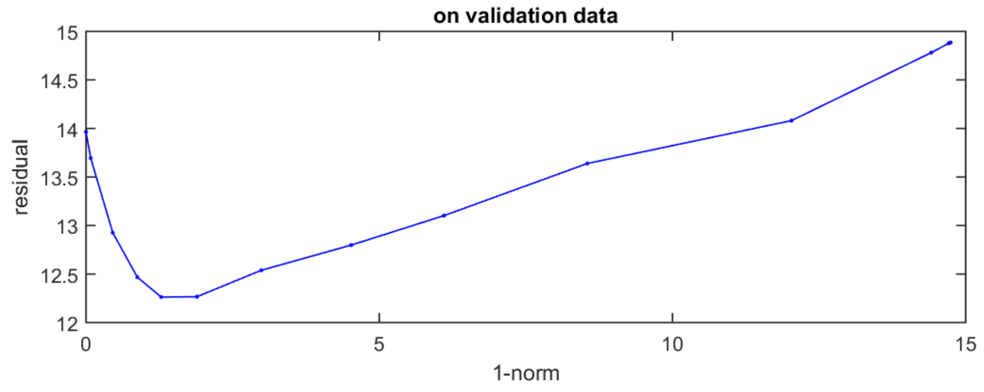
```
for i = 1:N
    disp(i)
    lambda = lam_vals(i);
    x = ista_solve(At,bt,lambda);
    bhat = sign(At*x);
    err(i) = mean( bhat ~= bt );
    res(i) = norm(At*x-bt,2);
    nrm(i) = norm( x, 1);   % size of 1-norm
    nonz(i) = sum( abs(x) > 1e-6 );   % number of nonzeros

    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('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('residual'); title('on validation data')
subplot(212); plot( nonz, errv, 'b.-')
xlabel('nonzeros'); ylabel('error rate')
```

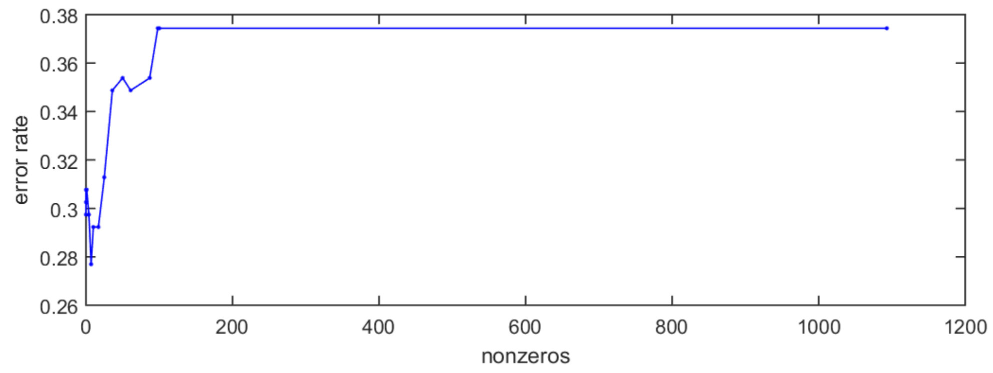
And here are the plots. The residual vs 1-norm looks convex as expected. And we see that the more we penalize the 1-norm, the more we encourage sparsity.



d) Repeat parts **b)** and **c)**. 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 code is very similar so we omit it. Here are the plots:





This time 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.

e) Compare the performance of the Lasso and Ridge Regression in this application. Do this by the following steps. Randomly split the set of 295 patients into 10 subsets of size 29-30. Use these subsets for training, tuning, and testing. Use the data in 8 subsets to find a solution $\widehat{\boldsymbol{\beta}}$ to the Lasso optimization above and to the ridge regression

$$\underset{\boldsymbol{\beta}\in\mathbb{R}^n}{\text{minimize}} \quad \|\boldsymbol{X}\boldsymbol{\beta}-\boldsymbol{y}\|_2^2+\lambda\|\boldsymbol{\beta}\|_2^2 \,.$$

Repeat this for a range of λ values, each yielding a solution $\widehat{\boldsymbol{\beta}}_\lambda$. Then compute the prediction error using each $\widehat{\boldsymbol{\beta}}_\lambda$ on one of the remaining two subsets. Select the $\widehat{\boldsymbol{\beta}}_\lambda$ that has the smallest prediction error (one for the Lasso and one for ridge regression). Finally, compute the test error on the final subset of data by comparing both the squared error and the count of how labels were incorrectly predicted, i.e., $\text{sign}(\boldsymbol{x}_i^T\widehat{\boldsymbol{\beta}}_\lambda)\neq y_i$. To get a better sense of the performances, 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 simply swap out the ISTA solver with either Landwebber (gradient descent), or by the analytic solution to the ridge regression problem. Note, in this scenario it is *much* more efficient to compute:

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

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, 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.