ECE 4803, Fall 2020

Homework #4

Due Tuesday, September 29, at 9:30am

1. Prepare a one paragraph summary of what we talked about in class since the last assignment. I do not want just a bulleted list of topics, I want you to use complete sentences and establish context (Why is what we have learned relevant? How does it connect with other classes?). The more insight you give, the better.

2. Recall the Tikhonov regularized least squares problem

$$\underset{\boldsymbol{x}\in\mathbb{R}^N}{\text{minimize}} \ \|\boldsymbol{y}-\boldsymbol{A}\boldsymbol{x}\|_2^2+\delta\|\boldsymbol{x}\|_2^2.$$

In this problem we will justify the formulas for the solution to this problem given on page 48 in the notes.

(a) Show that the Tikhonov regularized least squares problem can be rewritten in the format of a standard least squares problem

$$\underset{\boldsymbol{x}\in\mathbb{R}^N}{\text{minimize}} \ \|\widetilde{\boldsymbol{y}}-\widetilde{\boldsymbol{A}}\boldsymbol{x}\|_2^2$$

for an appropriate choice of $\widetilde{\boldsymbol{A}}$ and $\widetilde{\boldsymbol{y}}$.

(b) Derive the solution $\widehat{\boldsymbol{x}}=(\boldsymbol{A}^{\mathrm{T}}\boldsymbol{A}+\delta\mathbf{I})^{-1}\boldsymbol{A}^{\mathrm{T}}\boldsymbol{y}$ from the formula for the solution to the standard least squares problem together with the $\widetilde{\boldsymbol{A}}$ and $\widetilde{\boldsymbol{y}}$ computed in part (a). [Hint: a useful fact here is that if

$$\widetilde{\boldsymbol{A}}=\begin{bmatrix}\boldsymbol{A}\\ \boldsymbol{B}\end{bmatrix}$$

then

$$\widetilde{\boldsymbol{A}}^{\mathrm{T}}\widetilde{\boldsymbol{A}}=\boldsymbol{A}^{\mathrm{T}}\boldsymbol{A}+\boldsymbol{B}^{\mathrm{T}}\boldsymbol{B}.$$

You do not need to prove this, but you should convince yourself that it is true.]

(c) Derive the formula $\widehat{\boldsymbol{x}}=\boldsymbol{V}(\boldsymbol{\Sigma}^2+\delta\mathbf{I})^{-1}\boldsymbol{\Sigma}\boldsymbol{U}^{\mathrm{T}}\boldsymbol{y}$ by plugging $\boldsymbol{A}=\boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^{\mathrm{T}}$ into the formula in part (b). In this problem, assume that $R=N$.[1] [Hint: recall that if $\boldsymbol{A},\boldsymbol{B},\boldsymbol{C}$ are invertible matrices, then $(\boldsymbol{A}\boldsymbol{B}\boldsymbol{C})^{-1}=\boldsymbol{C}^{-1}\boldsymbol{B}^{-1}\boldsymbol{A}^{-1}$.]

(d) In the notes I also claimed that another formula for the solution to the Tikhonov regularized least squares problem was $\widehat{\boldsymbol{x}}=\boldsymbol{A}^{\mathrm{T}}(\boldsymbol{A}\boldsymbol{A}^{\mathrm{T}}+\delta\mathbf{I})^{-1}\boldsymbol{y}$. Show that by plugging $\boldsymbol{A}=\boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^{\mathrm{T}}$, you arrive at the same formula as in part (c). In this problem, assume that $R=M$.

[1] This result is actually true for any M, N, and R, but I'm only asking you to prove it in the full-rank case because it's a slightly simpler argument. The fact that this makes the argument is a hint. What extra fact about the SVD holds when $R=N$?

3. In the notes we showed that when using the truncated SVD, we could decompose the error that we incur as a result of noise into "noise error" and "approximation error" terms of the form

$$\widehat{\boldsymbol{x}}_{\text{trunc}} - \widehat{\boldsymbol{x}}_{\text{clean}} = \sum_{r=1}^{R'} \frac{1}{\sigma_r} \langle \boldsymbol{e}, \boldsymbol{u}_r \rangle \boldsymbol{v}_r + \sum_{r=R'+1}^{R} -\langle \boldsymbol{x}, \boldsymbol{v}_r \rangle \boldsymbol{v}_r.$$

Show that a similar decomposition is possible for Tikhonov regularization. Specifically, derive an expression of the form

$$\widehat{\boldsymbol{x}}_{\text{tik}} - \widehat{\boldsymbol{x}}_{\text{clean}} = \sum_{r=1}^{R} \underline{\qquad\quad} \langle \boldsymbol{e}, \boldsymbol{u}_r \rangle \boldsymbol{v}_r + \sum_{r=1}^{R} \underline{\qquad\quad} \langle \boldsymbol{x}, \boldsymbol{v}_r \rangle \boldsymbol{v}_r.$$

4. In this problem you will write Python functions that implement two versions the gradient descent for the least squares problem. Each function should take as input a generic matrix $\boldsymbol{A}$ and vector $\boldsymbol{y}$, an initial guess $\boldsymbol{x}^{(0)}$, and stopping criteria (I would include both a tolerance ϵ as well as a maximum number of iterations, just in case). While you wouldn't normally save the results of each iteration, to help you understand what is going on, your code should return an array containing the sequence of iterates $\boldsymbol{x}^{(0)}, \boldsymbol{x}^{(1)}, \ldots$.

As I said above, I would like you to implement two versions. One version should be what was described in the notes (you can implement either version – the problems we will be working on here are not going to be big enough for the computational difference to be noticeable). The other version should be gradient descent with α_k set to a fixed constant. In the context of least squares, we can work out analytically what a good value for each step size is, but we will see later that in many optimization problems, finding such an analytical solution may not be possible. Here we will see what kind of impact this can have. Note that your version with a fixed step size will need an additional input (the step size α).

(a) First let's test both versions on the simplest possible least squares problem I can think of

$$\boldsymbol{A} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \qquad \boldsymbol{y} = \begin{bmatrix} 1 \\ 1 \end{bmatrix}.$$

Hopefully just by looking at this problem you can see that the solution to

$$\underset{\boldsymbol{x} \in \mathbb{R}^N}{\text{minimize}} \ \|\boldsymbol{y} - \boldsymbol{A}\boldsymbol{x}\|_2^2$$

will just be $\widehat{\boldsymbol{x}} = \boldsymbol{y} = [1, 1]^{\mathrm{T}}$. Verify that this is indeed what both versions of your code produces when using an initial estimate of $\boldsymbol{x}^{(0)} = \boldsymbol{0}$. Report how many iterations are required for both versions, reporting the results for several different values of the step size in the fixed step size version. What step size gives you the best performance here?

(b) Let's make the problem a little more interesting and consider

$$\boldsymbol{A} = \begin{bmatrix} 1 & 0.99 \\ 0.99 & 1 \end{bmatrix} \qquad \boldsymbol{y} = \begin{bmatrix} 1.99 \\ 1.99 \end{bmatrix}.$$

It may not be *quite* as obvious this time, but it should be clear that in this case we still have a solution of $[1, 1]^{\mathrm{T}}$. Compute the condition number $\kappa(\boldsymbol{A})$. Based on our discussion in class, what does this tell us about how many iterations we might expect to need compared to the previous problem? Try out both versions of the algorithm and report how many iterations are required. If the answer is not what you expected, can you hypothesize why?

(c) Let's consider one more tweak of this problem:

$$\boldsymbol{A} = \begin{bmatrix} 1 & 0.99 \\ 1 & 1.01 \end{bmatrix} \qquad \boldsymbol{y} = \begin{bmatrix} 1.99 \\ 2.01 \end{bmatrix}.$$

The solution remains $[1,1]^{\mathrm{T}}$. Again, compute the condition number $\kappa(\boldsymbol{A})$, and try out both versions of the algorithm, reporting how many iterations are required. What is different from the previous problem? (If you are struggling to explain what is happening, it may help to plot the iterates together with a contour plot of the least squares objective function, and/or to experiment with different starting points for $\boldsymbol{x}^{(0)}$.

(d) In part (c), you may need to set the tolerance parameter ϵ to be very small in order to avoid having the algorithm terminate at a point $\boldsymbol{x}^{(k)}$ where $\boldsymbol{x}^{(k)}$ is still quite far from the true solution. Explain why this is necessary?

5. Gradient descent can also be used to solve Tikhonov regularized least squares problems. Consider the least squares problem defined by

$$\boldsymbol{A} = \begin{bmatrix} 10000 & 10001 \\ 10001 & 10002 \\ 10002 & 10003 \\ 10003 & 10004 \\ 10004 & 10005 \end{bmatrix} \qquad \boldsymbol{y} = \begin{bmatrix} 20001 \\ 20003 \\ 20005 \\ 20007 \\ 20009 \end{bmatrix}.$$

This is a somewhat ridiculous generalization of the previous example from part (c) of the previous problem, and the optimal solution to the associated least squares problem is again $\boldsymbol{x}^\star = [1,1]^{\mathrm{T}}$

(a) What is the condition number $\kappa(\boldsymbol{A})$? Do you think we should even try to apply our gradient descent code to this?

(b) Do it anyway. Using the version of your code that automatically adapts the step size, try running your code and comment on what happens. Does it converge? If so, to what? If not, what is the output when the iteration limit is reached?

(c) Now consider the Tikhonov regularized version of the problem. One way to implement a gradient descent solver for this problem is by treating it as a modified version of least squares as in problem 2. Do this for a few values of δ. For each, report the condition number of $\widetilde{\boldsymbol{A}}$, the number of iterations required for convergence, and the error $\|\widehat{\boldsymbol{x}} - \boldsymbol{x}^\star\|_2$. What value of δ works best?

(d) Compare the results from Tikhonov regularization to the truncated SVD. (Instead of the regularization parameter δ, you must now choose the truncation level R', but here you don't have a lot of wiggle room for this). In this case, which approach seems to work better? Can you explain why? (Think about the relationship between $\boldsymbol{x}^\star$ and the singular vectors of $\boldsymbol{A}$.)