Math Foundations of ML, Fall 2017

Homework #7

Due Monday October 30, at the beginning of class

As stated in the syllabus, unauthorized use of previous semester course materials is strictly prohibited in this course.

1. Using you class notes, prepare a 1-2 paragraph summary of what we talked about in class in the last week. 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 things you have learned here or in other classes?). The more insight you give, the better.

2. Let
$$\boldsymbol{H} = \begin{bmatrix} 2 & 1 \\ 1 & 2 \end{bmatrix}, \quad \boldsymbol{b} = \begin{bmatrix} -1 \\ -3 \end{bmatrix},$$
and let $f : \mathbb{R}^2 \to \mathbb{R}$ be
$$f(\boldsymbol{x}) = \frac{1}{2}\boldsymbol{x}^{\mathrm{T}}\boldsymbol{H}\boldsymbol{x} - \boldsymbol{b}^{\mathrm{T}}\boldsymbol{x}.$$
   (a) What is the smallest value that f takes on $\mathbb{R}^2$? At what $\hat{\boldsymbol{x}}$ does it achieve this minimum value?
   (b) Write $f(\boldsymbol{x})$ out as a quadratic function in x_1, x_2 where $\boldsymbol{x} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$. In other words, fill in the blanks below
$$f(\boldsymbol{x}) = \_\_x_1^2 + \_\_x_2^2 + \_\_x_1x_2 + \_\_x_1 + \_\_x_2.$$
   (c) Using MATLAB or Python, make a contour plot of $f(\boldsymbol{x})$ around it's minimizer in $\mathbb{R}^2$.
   (d) On top of the contour plot, trace out the first four steps of the gradient descent algorithm starting at $\boldsymbol{x}_0 = \mathbf{0}$.
   (e) On top of the contour plot, trace out the two steps of the conjugate gradient method starting at $\boldsymbol{x} = \mathbf{0}$.

3. Write a MATLAB function `sdsolve` that implements the steepest descent algorithm. The function should be called as

```
[x, iter] = sdsolve(H, b, tol, maxiter);
```

where `H` is a $N \times N$ symmetric positive definite matrix, `b` is a vector of length N, and `tol` and `maxiter` specify the halting conditions. Your algorithm should iterate until $\|r_k\|_2/\|b\|_2$ is less than `tol` or the maximum number of iterations `maxiter` has been reached. For the outputs: `x` is your solution, and `iter` is the number of iterations that were performed. Run your code on the $\boldsymbol{H}$ and $\boldsymbol{b}$ in the file `hw7p3_data.mat` for a `tol` of 10^{-6}. Report the number of iterations needed for convergence, and for your solution $\hat{\boldsymbol{x}}$ verify that $\boldsymbol{H}\hat{\boldsymbol{x}}$ is within the specified tolerance of $\boldsymbol{b}$.

4. Write a MATLAB function `cgsolve.m` that implements the method of conjugate gradients. The function should be called as

```
[x, iter] = cgsolve(H, b, tol, maxiter);
```

where the inputs and outputs have the same interpretation as in the previous problem. Run your code on the $\boldsymbol{H}$ and $\boldsymbol{b}$ in the file `hw7p3_data.mat` for a `tol` of 10^{-6}. Report the number of iterations needed for convergence, and for your solution $\hat{\boldsymbol{x}}$ verify that $\boldsymbol{H}\hat{\boldsymbol{x}}$ is within the specified tolerance of $\boldsymbol{b}$.

5. In this question, we will explore how the kernel regression problem can be solved using online least-squares. Data points $(\boldsymbol{t}_1, y_1), (\boldsymbol{t}_2, y_2), \ldots$ are streaming in ... suppose that after observing the first m samples, we have formed the estimate of the optimal coefficients

$$\hat{\alpha}_m = (\boldsymbol{K}_m + \delta\mathbf{I})^{-1}\boldsymbol{y}_m, \quad \text{where} \quad K_m[i,j] = k(\boldsymbol{t}_j, \boldsymbol{t}_i), \quad \boldsymbol{y}_m = \begin{bmatrix} y_1 \\ \vdots \\ y_m \end{bmatrix},$$

and have stored the $m \times m$ matrix $\boldsymbol{P}_m = (\boldsymbol{K}_m + \delta\mathbf{I})^{-1}$. Now sample $(\boldsymbol{t}_{m+1}, y_{m+1})$ appears, and we want to compute $\hat{\boldsymbol{\alpha}}_{m+1} = (\boldsymbol{K}_{m+1} + \delta\mathbf{I})^{-1}\boldsymbol{y}_{m+1}$ ...

(a) Show how the $m+1 \times m+1$ matrix

$$\boldsymbol{P}'_{m+1} = \begin{bmatrix} \boldsymbol{K}_m + \delta\mathbf{I} & \mathbf{0} \\ \mathbf{0} & \delta \end{bmatrix}^{-1}$$

can be (efficiently) computed from $\boldsymbol{P}_m$.

(b) Show how the $m+1 \times m+1$ matrix $\boldsymbol{P}_{m+1}$ can be written using a low-rank update of $\boldsymbol{P}'_{m+1}$. That is, find $m+1 \times R$ matrices $\boldsymbol{B}, \boldsymbol{C}$ such that

$$\boldsymbol{P}_{m+1} = \left(\boldsymbol{P}'^{-1}_{m+1} + \boldsymbol{B}\boldsymbol{C}^{\mathrm{T}}\right)^{-1},$$

and then apply the matrix inversion lemma to show how the key to the calculation above is solving an $R \times R$ system of equations. (As we discussed in class, you should be able to do this with $R = 2$.)

(c) Write the pseudo-code (similar to what is on page II.134 of the notes) for online kernel regression.

(d) Suppose that we only want our estimate to depend on the last Q samples. That is, after observing $m > Q$ samples, we want to base our estimate on $(\boldsymbol{t}_{m-Q+1}, y_{m-Q+1}), \ldots, (\boldsymbol{t}_m, y_m)$. Then when we observe sample $(\boldsymbol{t}_{m+1}, y_{m+1})$, we first *remove* the effect of sample $(\boldsymbol{t}_{m-Q+1}, y_{m-Q+1})$ from our estimate, then *add* the effect of $(\boldsymbol{t}_{m+1}, y_{m+1})$. Describe in detail how this can be accomplished using low rank updates via the matrix inversion lemma.