Indian Institute of Technology Kanpur
CS771 Introduction to Machine Learning

Instructor: Purushottam Kar
Date: April 04, 2018
Total: 60 + bonus marks

ASSIGNMENT 3

Instructions

1. Only electronic submissions to Gradescope `https://gradescope.com` will be accepted. No hard copy or emailed submissions would be accepted.
2. Your submission should be a single PDF file compiled using the LaTeX style file `saltsubmit.sty`.
3. No zip/tar/png/jpg files or handwritten and then scanned submissions will be accepted.
4. Submissions will be accepted till April 20, 2018, 2359 hrs, IST.
5. **There will be no separate late submission deadline for this assignment**. Solutions will be released soon after the regular deadline.
6. We will be closely checking your submissions for instances of plagiarism.
7. You may be penalized if you do not follow the formatting instructions carefully.
8. Your answer to every question should begin on a new page. The style file is designed to do this automatically. However, if it fails to do so, use the `\clearpage` option in LaTeX before starting the new question, to enforce this.
9. An account has been created for you on Gradescope. Use your IITK CC ID (not GMail, CSE etc IDs) to login and use the "Forgot Password" option to set your password initially.
10. While submitting your assignment on this website, you will have to specify on which page(s) is question 1 answered, on which page(s) is question 2 answered etc. To do this properly, first ensure that the answer to each question starts on a different page.
11. Be careful to flush all your floats (figures, tables) corresponding to question n before starting the answer to question $n+1$ otherwise graders might miss your figures and award you less points. Remember, different people may grade question n and $n+1$.
12. Your solutions must appear in proper order in the PDF file i.e. your solution to question n must be complete in the PDF file (including all plots, tables, proofs etc) before you present a solution to question $n+1$.

Problem 3.1 (MajMin to the Rescue). A curious but useful family of functions that arise in learning tasks is the so-called DC family. These functions are expressable as $f(\mathbf{x}) = g(\mathbf{x}) - h(\mathbf{x})$ where g, h are both convex functions, hence the name DC (difference of convex). Despite their simple structure, this family is ridiculously powerful. Apart from popping up in ranking and detection tasks, it can be shown that *every* twice continuously differentiable function is DC!

Two popular algorithms used to minimize DC functions (unconstrained for now) are the DCA (DC Algorithm) and the CCCP (Convex-ConCave Procedure), both given below.

```
Algorithm 1: DCA

Input: A DC function $f = g - h$
1: $\mathbf{x}^0 \leftarrow$ INIT                    //Initialize
2: for $t = 0, 1, 2, \ldots,$ do
3:     Let $\mathbf{z}^{t+1} \leftarrow \nabla h(\mathbf{x}^t)$
4:     Let $\mathbf{x}^{t+1} \leftarrow \arg\min_{\mathbf{x}} \; g(\mathbf{x}) - \langle \mathbf{z}^{t+1}, \mathbf{x} \rangle$
5: end for
```

```
Algorithm 2: CCCP

Input: A DC function $f = g - h$
1: $\mathbf{x}^0 \leftarrow$ INIT                    //Initialize
2: for $t = 0, 1, 2, \ldots,$ do
3:     Let $\mathbf{z}^{t+1} \leftarrow \nabla h(\mathbf{x}^t)$
4:     Let $\mathbf{x}^{t+1}$ so that $\nabla g(\mathbf{x}^{t+1}) = \mathbf{z}^{t+1}$
5: end for
```

1. Show that every quadratic function of the type $f(\mathbf{x}) = c + \langle \mathbf{b}, \mathbf{x} \rangle + \frac{1}{2}\mathbf{x}^\top A\mathbf{x}$, with $c \in \mathbb{R}, \mathbf{x}, \mathbf{b} \in \mathbb{R}^d$ and $A \in \mathbb{R}^{d\times d}$ a symmetric matrix, is a DC function.
2. Show that DCA simply implements the Majorization Minimization algorithm.
3. Show that DCA and CCCP are the same algorithm. Assume g, h are differentiable.
4. What should CCCP conclude if there doesn't exist any $\mathbf{x}^{t+1}$ so that $\nabla g(\mathbf{x}^{t+1}) = \nabla h(\mathbf{x}^t)$?

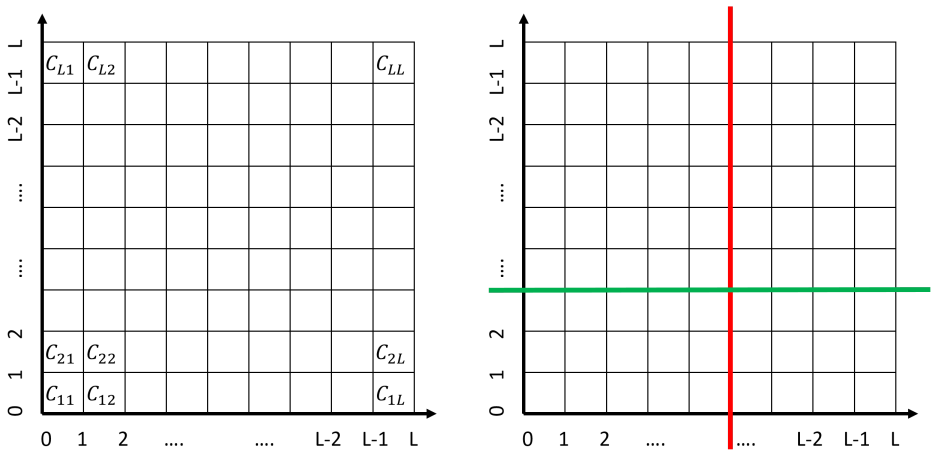
Hint: For this problem you may need to use the Danskin's theorem. A description of the result is available at `en.wikipedia.org/wiki/Danskin's_theorem`. (3+2+15+5=25 marks)

Problem 3.2 (A single pixel Camera?). A modern digital camera captures an image by receiving light rays onto an array of pixels. For sake of simplicity, assume that there are d pixels are arranged in a row (in practice they are arranged in a matrix but this assumption is convenient) and that each pixel $p_i, i \in [d]$ captures a real valued intensity $x_i \in \mathbb{R}$ based on the amount of light falling on that pixel. Thus, an image is represented as a d-dimensional vector $\mathbf{x} \in \mathbb{R}^d$.

However, since typically $d \gg 1$, the cameras internally perform a Fourier (or DCT) transform and retain only a few leading components of the transform, thus throwing away a huge chunk of (albeit less useful) information all those pixels had gathered. The Fourier transform is represented as an orthonormal transformation $F \in \mathbb{R}^{d\times d}$ i.e. the Fourier transform can be obtained as $\hat{\mathbf{x}} = F\mathbf{x}$ and $FF^\top = F^\top F = I_{d\times d}$. Can we avoid this wastage of information using sparse recovery techniques? It turns out that we only need a single pixel to do this!

Our single pixel camera is designed as follows. It consists of $k \ll d$ *masks* $\mathbf{a}_1, \ldots, \mathbf{a}_k \in \mathbb{R}^d$ that are fixed at the time the camera is manufactured. When taking a photograph of an image $\mathbf{x} \in \mathbb{R}^d$, we simply store the values $\mathbf{y} = (\langle \mathbf{a}_1, \mathbf{x} \rangle, \ldots, \langle \mathbf{a}_k, \mathbf{y} \rangle) \in \mathbb{R}^k$. Notice that only k light capturing pixels are required to do this (in fact, even a single one can be shown to suffice). We are capturing less information and since pixels are expensive, this camera may cost less too!

Design an algorithm to first come up with the masks, and then given the masks, design an algorithm to retrieve the image $\mathbf{x}$ from the captured data $\mathbf{y}$. Prove that your algorithm (under necessary assumptions) does recover the image (near) perfectly. For your analysis, assume that we are only interested in images with sparse Fourier transforms i.e. $\|\hat{\mathbf{x}}\|_0 \leq s \ll d$. However, be careful that the images themselves will not be sparse i.e. we may have $\|\mathbf{x}\|_0 \approx d$. You may also find reference [**JAK**] Theorem 7.1 useful for your problem. (20 marks)



Problem 3.3 (Overcomplicated Objectives). Let $f, g : \mathbb{R}^d \rightarrow \mathbb{R}$ be two convex functions. Design and analyze the convergence of an algorithm for the following optimization problem.

$$\min_{\mathbf{x} \in \mathbb{R}^d} \max \{f(\mathbf{x}), g(\mathbf{x})\}$$

Clearly state any assumptions on f, g and any known results you are using. (15 marks)

Problem 3.4 (**BONUS** A Tale of Two Tracts of Land – Part II). Recall the last problem in the mid-semester examination. When the brothers Alex and Bob actually performed the test suggested by you, they discovered that their mother Dana had indeed made a mistake while allocating the chunks and $\Delta \approx 0.1 \ll 0.5$ i.e. Bob's chunks together give him a much higher total productivity than those left to Alex.

Now the brothers have to reallocate land among themselves. To do so in an amicable manner they decide the following. They represent their square field as a 2D square $[0, L] \times [0, L]$ and re-chunk it so that each chunk is now a tiny axis-aligned square of unit side length (see figure). There are a total of L^2 chunks in the field $C_{11}, \ldots, C_{1L}, C_{21}, \ldots, C_{2L}, \ldots, C_{LL}$ i.e $N = L^2$.

The brothers also decide that they are going to partition the field only using axis aligned lines so that no chunk is split i.e. the partition between Alex and Bob is either going to be of the form $x = m$ for some $m \in [L]$ or of the form $y = n$ for some $n \in [L]$. The red and green lines in the figure illustrate two such possible partitions. As before, the brothers cannot test all the chunks $C_{ij}, i, j \in [L]$ for their productivity $\alpha(C_ij)$ and can only test $n \ll N = L^2$ chunks.

Design and analyze an algorithm to find the most fair axis aligned partition of the chunks i.e. the partition for which $|\Delta - 0.5|$ is the smallest. Note that no axis aligned partition may turn out to be very fair – we just want to find the most fair one. (25 marks)