Indian Institute of Technology Kanpur
CS771 Introduction to Machine Learning

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

ASSIGNMENT 3

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 expressible 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)

Solution. We give the solutions in parts below

1. First, notice that if p, q are two DC functions and $a, b \in \mathbb{R}$ then the function $a \cdot p + b \cdot q$ is also DC. Then notice that linear and affine functions (for example $c + \langle \mathbf{b}, \mathbf{x} \rangle$) are both convex as well as concave. Next, notice that

$$\frac{1}{2}\mathbf{x}^\top A \mathbf{x} = \sum_i \sum_j \frac{A_{ij}}{2} \cdot \mathbf{x}_i \mathbf{x}_j$$

Thus, if we prove that for arbitrary $i, j \in [n]$, the function $p(x, y) = xy$ (where $x, y \in \mathbb{R}$) is DC then we would have proved that f is DC. However, this is easy to see since $xy = \frac{(x+y)^2 - (x-y)^2}{4}$ and thus, a DC function. Note that whereas it seems tempting to prove this result via an eigenvalue decomposition, that requires A to be diagonalizable and is a slightly more tedious route. The above proof works for any matrix A, even those that are non-symmetric and non-diagonalizable.

2. Since $h(\mathbf{x})$ is convex, we have, for any $\mathbf{x}$

$$h(\mathbf{x}) \geq h(\mathbf{x}^t) + \left\langle \nabla h(\mathbf{x}^t), \mathbf{x} - \mathbf{x}^t \right\rangle,$$

which gives us

$$f(\mathbf{x}) \leq g(\mathbf{x}) - h(\mathbf{x}^t) - \left\langle \nabla h(\mathbf{x}^t), \mathbf{x} - \mathbf{x}^t \right\rangle = g(\mathbf{x}) - \left\langle \nabla h(\mathbf{x}^t), \mathbf{x} \right\rangle + \left\langle \nabla h(\mathbf{x}^t), \mathbf{x}^t \right\rangle - h(\mathbf{x}^t)$$

Also notice that the left and right hand sides are equal at $\mathbf{x}^t$. Thus, the right hand side is a proper majorization of the function f. Now notice that the DCA algorithm minimizes the right hand side at each time step by minimizing $g(\mathbf{x}) - \left\langle \nabla h(\mathbf{x}^t), \mathbf{x} \right\rangle$. Thus, DCA does implement the majorization minimization procedure.

3. Note that $g(\mathbf{x}) = \max_{\mathbf{z}} \ \langle \mathbf{z}, \mathbf{x} \rangle - g^*(\mathbf{z})$ where $g^*(\mathbf{z}) = \max_{\mathbf{x}} \ \langle \mathbf{z}, \mathbf{x} \rangle - g(\mathbf{x})$ is the Fenchel dual of g. For any $\mathbf{x}$, let $\tilde{\mathbf{z}}$ be such that $g(\mathbf{x}) = \langle \tilde{\mathbf{z}}, \mathbf{x} \rangle - g^*(\tilde{\mathbf{z}})$. Since g is differentiable, we are assured by Danskin's theorem that $\nabla g(\mathbf{x}) = \tilde{\mathbf{z}}$ and that $\tilde{\mathbf{z}}$ is unique.

CCCP ensures $g^*(\mathbf{z}^{t+1}) = \left\langle \mathbf{z}^{t+1}, \mathbf{x}^{t+1} \right\rangle - g(\mathbf{x}^{t+1})$ since $\mathbf{x}^{t+1} \leftarrow \arg\max_{\mathbf{x}} \ \left\langle \mathbf{z}^{t+1}, \mathbf{x} \right\rangle - g(\mathbf{x})$ which is a restatement of $\mathbf{x}^{t+1} \leftarrow \arg\min_{\mathbf{x}} \ g(\mathbf{x}) - \left\langle \mathbf{z}^{t+1}, \mathbf{x} \right\rangle$. By the uniqueness guarantee given above, we are assured that $\nabla g(\mathbf{x}^{t+1}) = \mathbf{z}^{t+1}$.

4. Since the two algorithms are identical, the only case when CCCP fails to find a solution would be when DCA also fails to find an optimum for $g(\mathbf{x}) - \left\langle \nabla h(\mathbf{x}^t), \mathbf{x} \right\rangle$. This can happen when the function is unbounded from below. However, since DCA always minimizes an upper bound on f, this means that f itself is unbounded from below i.e. $\min_{\mathbf{x}} \ f(\mathbf{x}) = -\infty$.

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 $\mathbf{y} = (\langle \mathbf{a}_1, \mathbf{x} \rangle, \ldots, \langle \mathbf{a}_k, \mathbf{x} \rangle) \in \mathbb{R}^k$. 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)

Algorithm 3: Single Pixel Camera

Input: Size of image d pixels, spectral sparsity s

1: Let $k \leftarrow 25Cs \log \frac{d}{s}$
2: **for** $i = 1, 2, \ldots, k$ **do**
3: Let $\mathbf{p}_i \leftarrow \mathcal{N}(\mathbf{0}, \frac{1}{k} \cdot I)$
4: Let $\mathbf{a}_i = F^\top \mathbf{p}_i$
5: **end for**
6: Acquire image $\mathbf{x}$ as $\mathbf{y} = (\langle \mathbf{a}_1, \mathbf{x} \rangle, \ldots, \langle \mathbf{a}_k, \mathbf{x} \rangle) \in \mathbb{R}^k$
7: Solve $\hat{\mathbf{z}} = \arg\min_{\|\mathbf{z}\|_0 \leq s} \sum_{i=1}^k (\langle \mathbf{p}_i, \mathbf{z} \rangle - \mathbf{y}_i)^2$
8: Output $\tilde{\mathbf{x}} = F^\top \mathbf{z}$

Solution. Let $\mathbf{x}$ be the image to be captured and let $\hat{\mathbf{x}} = F\mathbf{x}$ be its (s-sparse) Fourier transform. Denote $P \in \mathbb{R}^{k \times d}$ as the matrix in Algorithm 3 each of whose rows is a scaled normal d-dimensional vector and let $A = PF$ where F is the Fourier transform matrix. Now, the algorithm captures $\mathbf{y} = A\mathbf{x} = PF\mathbf{x} = P\hat{\mathbf{x}}$.

Now if the matrix P satisfies the α-RSC and β-RSS conditions with respect to s-sparse vectors with a condition number $\kappa = \frac{\beta}{\alpha} < 2$, then we have seen in class (lecture 17) that using projected gradient descent to solve step 7 in Algorithm 3 will return, within $\mathcal{O}\left(\log \frac{1}{\epsilon}\right)$ steps, an s-sparse estimate $\hat{\mathbf{z}}$ that satisfies $\|P\hat{\mathbf{z}} - \mathbf{y}\|_2^2 \leq \epsilon$.

This means that $\|P\hat{\mathbf{z}} - P\hat{\mathbf{x}}\|_2^2 \leq \epsilon$ (recall that $\mathbf{y} = P\hat{\mathbf{x}}$). However, the RSC condition guarantees (since $\nabla_{\mathbf{z}} \ \|P\mathbf{z} - \mathbf{y}\|_2^2 = \mathbf{0}$ at $\mathbf{z} = \hat{\mathbf{x}}$), that $\|P(\hat{\mathbf{z}} - \hat{\mathbf{x}})\|_2^2 \geq \frac{\alpha}{2} \|\hat{\mathbf{z}} - \hat{\mathbf{x}}\|_2^2$. This gives us $\|\hat{\mathbf{z}} - \hat{\mathbf{x}}\|_2^2 \leq \frac{2\epsilon}{\alpha}$. Since the Fourier transform is an orthonormal transformation, it has a unit condition number. Thus, $\|\tilde{\mathbf{x}} - \mathbf{x}\|_2^2 = \left\|F^\top \hat{\mathbf{z}} - \mathbf{x}\right\|_2^2 = \left\|F^\top (\hat{\mathbf{z}} - \hat{\mathbf{x}})\right\|_2^2 = \|\hat{\mathbf{z}} - \hat{\mathbf{x}}\|_2^2 \leq \frac{2\epsilon}{\alpha}$.

All that remains to be shown is that the matrix P does satisfy the RSC/RSS conditions. Let $\mathbf{r} \sim \mathcal{N}(\mathbf{0}, I) \in \mathbb{R}^d$ be a standard d-dimensional Normal random vector. Then it is easy to see that for any (fixed) unit vector $\mathbf{v} \in \mathbb{R}^d$ we have $\langle \mathbf{r}, \mathbf{v} \rangle \sim \mathcal{N}(0, 1)$ i.e. the inner product is distributed as a standard scalar normal variable.

Let $R \in \mathbb{R}^{k \times d}$ be a matrix, each of whose rows is a standard normal vector i.e. $\mathbf{r}_i \sim \mathcal{N}(\mathbf{0}, I)$. Then the above result shows that $\|R\mathbf{v}\|_2^2 \sim \chi^2(k)$ i.e. the squared norm of the vector $R\mathbf{v}$ is distributed as a standard Chi-squared distribution with k degrees of freedom. It is easy to prove that $\mathbb{E}\left[\|R\mathbf{v}\|_2^2\right] = k$. Moreover, standard results[1] for subexponential variables tell us that

$$\mathbb{P}\left[\left|\|R\mathbf{v}\|_2^2 - k\right| > t\right] \leq 2\exp\left(-\frac{t^2}{24ke^2}\right)$$

for all $t < 2e\sqrt{3} \cdot k$. Taking $t = \epsilon \cdot k$ for any $\epsilon < 1$ gives us.

$$\mathbb{P}\left[\left|\|R\mathbf{v}\|_2^2 - k\right| > \epsilon \cdot k\right] \leq 2\exp\left(-\frac{k\epsilon^2}{24e^2}\right)$$

Thus, the matrix $P = \frac{1}{\sqrt{k}} \cdot R$ satisfies the requirements of [**JAK**] Theorem 7.1. We will let C denote the constant in the Big Omega notation in that result. Thus, if we take $k = 25Cs \log \frac{d}{s}$, then this will ensure that the matrix P has an RIP constant of at most 0.2 with respect to s-sparse vectors and a restricted condition number of at most 1.5. Similar results also hold[2] if the matrix R has Rademacher entries or even subGaussian entries instead of normal entries.

[1] See for example Roman Vershynin. Introduction to the non-asymptotic analysis of random matrices. Lemmata 5.5, 5.15, Proposition 5.17

[2] See for example Dimitris Achlioptas. Database-friendly random projections, PODS 2001.

Problem 3.3 (Overcomplicated Objectives)**.** Let $f, g : \mathbb{R}^d \to \mathbb{R}$ 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)

Solution. We will employ a solution based on Danskin's theorem. We will assume f and g are differentiable as well as have bounded gradients i.e. $\max\{\|\nabla f(\mathbf{x})\|_2, \|\nabla g(\mathbf{x})\|_2\} \leq G$ for all $\mathbf{x} \in \mathbb{R}^d$. Let $h(\mathbf{x}) = \max\{f(\mathbf{x}), g(\mathbf{x})\}$. Then, by applying Danskin's theorem it is clear that in Algorithm 4, we have $\mathbf{g}^t \in \partial h(\mathbf{x}^t)$ at all times t. Thus, Algorithm 4 is performing sub-gradient descent with respect to the function h and our classroom results (lecture 15) tell us that we have $h(\hat{\mathbf{x}}^T) \leq h(\mathbf{x}^*) + \frac{\|\mathbf{x}^*\|_2^2 + G^2}{2\sqrt{T}}$ for any $\mathbf{x}^* \in \mathbb{R}^d$. In particular, the above holds also for all minimizers of h i.e. for all $\mathbf{x}^* \in \arg\min\ h(\mathbf{x})$.

Algorithm 4: MinMax Programming

Input: Convex differentiable functions f, g

Let $\mathbf{x}^0 \leftarrow \mathbf{0}$ //`Initialize`
for $t = 0, 1, 2, \ldots, T-1$ **do**
  if $f(\mathbf{x}^t) > g(\mathbf{x}^t)$ **then**
    $\mathbf{g}^t \leftarrow \nabla f(\mathbf{x}^t)$
  else
    $\mathbf{g}^t \leftarrow \nabla g(\mathbf{x}^t)$
  end if
  Let $\mathbf{x}^{t+1} \leftarrow \mathbf{x}^t - \frac{1}{\sqrt{T}} \cdot \mathbf{g}^t$
end for
return $\hat{\mathbf{x}}^T = \frac{1}{T}\sum_{t=1}^{T} \mathbf{x}^t$