CPSC 536M

Marcus Lai

November 2024

1 Introduction

In this project, I investigated subgradient methods and their variations since I was curious how non-differentiable, convex problems could be resolved. I analyze convergence proofs and report some results I've learned (Section 1 and 2) [1][2] . Some of the methods are implemented in Python and used to solve the task assignment problem to help improve understanding of tradeoffs of the variations and tuning(Section 3)[3][4]. The code for the report can be found here.

2 The Subgradient Descent

Subgradient methods apply to problems where the gradient could be unknown or difficult to compute.

Definition 2.1 (Subgradient)**.** A vector g is a subgradient of a function $f : \mathbf{R}^n \to \mathbf{R}$ if

$$f(y) \geq f(x) + g^T(y - x)$$

for all $y \in \mathbf{R}^n$.

Intuitively, this definition can be visualized as how much a hyperplane at $\mathbf{x}$ could tilt and still support the epigraph of f.

Similar to gradient descent, subgradient methods are first order methods. However, even when the subgradient condition is satisfied, there is no guarantee that the chosen direction will cause the objective to descend. Furthermore, the lack of a gradient also means step sizes cannot be reliably chosen. Having non-decreasing iterations makes subgradient descent slower than second order Newton methods and even gradient descent. Indeed we will see when the objective is differentiable, subgradient descent *is* gradient descent. Despite this, subgradient descent is vastly flexible and readily applicable to various problems, making it of particular interest.

The set of subgradients of f at $\mathbf{x}$ is the *subdifferential*, denoted $\delta f(\mathbf{x})$. The basic subgradient descent method at iteration k relies on choosing a subgradient vector, $g^{(k)} \in \delta f(x^{(k)})$, a step size $\alpha^{(k)} > 0$, and setting

$$x^{(k+1)} = x^{k+1} - a^{(k)} g^{(k)}.$$

We demonstrate that in the case that f is differentiable, Subgradient Descent reduces to the Gradient Descent since $\delta f(x)$ becomes the singleteon set $\{\nabla f(x)\}$.

Proposition 2.2 (Gradient Descent as a Special Case of Subgradient Descent)**.** *Suppose $f : \mathbf{R}^n \to \mathbf{R}$ is a differentiable function. Then the subgradient descent method is equivalent to the gradient descent method. Namely, the only possible choice for $g \in \delta f(x)$ is $\nabla f(x)$.*

Proof. It is clear that the gradient satisfies the subgradient property by virtue of convex functions. So choose $g' \in \delta f(x)$. Then

$$f(x) + \alpha g^T d \leq f(x + \alpha d) = f(x) + \alpha \nabla f(x)^T d + o(|\alpha|), \forall \alpha \in \mathbf{R}, d \in \mathbf{R^n}.$$

We set d be the direction $\nabla f(x) - g$ and obtain

$$\alpha(g^T \nabla f(x) - ||g||^2) \leq \alpha(||\nabla f(x)||^2 - \nabla f(x)^T g) + o(|\alpha|)$$

$$0 \leq \alpha ||\nabla f(x) - g||^2 + o(|\alpha|)$$

$$0 \leq ||\nabla f(x) - g||^2 \leq -o(|\alpha|)/\alpha \text{ for } \alpha < 0$$

By taking $\alpha \to 0$, we see $\nabla f(x) = g$ as required. □

2.1 Optimality Condition

Suppose $0 \in \delta f(x^*)$ for some x^*. Then by definition,

$$\forall y \in \mathbf{R^n}, f(y) \geq f(x^*) + \mathbf{0}^T(y - x^*) \iff f(y) \geq f(x^*).$$

Indeed $\mathbf{0}$ is in the subdifferential of f at x^* if and only if x^* is a minimizer of f, achieving optimality. This optimality condition yields a formal proof of a familiar result.

Proposition 2.3. *For a differentiable function f minimized over a non-empty convex set C, the following have the same minimizer:*

1. $\min_{x \in C} f(x)$
2. $\min_{x \in R^n} f(x) + I_C(x)$

where $I_C(x)$ is the indicator function ($I_C(x) = 0$ if $x \in C$ otherwise ∞).

Proof. First, we derive the subdifferential of $I_C(x)$. We proceed by definition to get $g \in \delta I_C(x)$ if

$$I_C(y) \geq I_C(x) + g^T(y - x), \ \forall y \in \mathbf{R^n}.$$

We note that this condition is always satisfied when $y \notin C$ and never satisfied when $x \notin C$ since C is nonempty. If $x \in C$ and $y \in C$, we have

$$0 \geq g^T(y - x) \iff g^T x \geq g^T y, \ \forall y \in C.$$

Now, notice that the set of gs that satisfy this is precisely the normal cone $N_C(x)$. So $\delta I_C(x) = N_C(x)$. Then,

$$\begin{aligned} \mathbf{0} \in \delta(f(x) + I_C(x)) \iff & \mathbf{0} \in \{\nabla f(x) + N_C(x)\} \\ \iff & -\nabla f(x) \in N_C(x) \\ \iff & -\nabla f(x)^T x \geq -\nabla f(x)^T y, \ \forall y \in C \\ \iff & \nabla f(x)^T (y - x) \geq 0, \ \forall y \in C \end{aligned}$$

the last line is the well known **first order optimiality condition** for f. So 1. is optimal at some point x if and only if 2. is optimal at x as required. □

2.2 Convergence and Choice of Stepsize

Unlike gradient descent, we do not have access to the gradient to perform procedures like line search to obtain an analytical guarantee that our objectives will descend at every iteration. However, by making certain assumptions, we can guarantee global and eventual convergence to within some constant of minimum, and even total convergence by some choices of step size. We consider the Euclidean distance between our kth iteration and some optimal minimizer x^*.

$$\begin{aligned} & ||\mathbf{x^{(k)}} - \mathbf{x}^*||^2 && (1) \\ =& ||\mathbf{x^{(k-1)}} - \alpha^{(k-1)}\mathbf{g^{(k-1)}} - \mathbf{x}^*||^2 && (2) \\ =& ((\mathbf{x^{(k-1)}} - \mathbf{x}^*) - \alpha^{(k-1)}\mathbf{g^{(k-1)}})^T((\mathbf{x^{(k-1)}} - \mathbf{x}^*) - \alpha^{(k-1)}\mathbf{g^{(k-1)}}) && (3) \\ =& ||\mathbf{x^{(k-1)}} - \mathbf{x}^*||^2 - 2\alpha^{(k-1)}(\mathbf{g^{(k-1)}})^T(\mathbf{x^{(k-1)}} - \mathbf{x}^*) + (\alpha^{(k-1)})^2||\mathbf{g^{(k-1)}}||^2 && (4) \\ \leq& ||\mathbf{x^{(k-1)}} - \mathbf{x}^*||^2 - 2\alpha^{(k-1)}(f(\mathbf{x^{(k-1)}}) - f(\mathbf{x}^*)) + (\alpha^{(k-1)})^2||\mathbf{g^{(k-1)}}||^2 && (5) \\ & (\text{... apply recursively ...}) && (6) \\ \leq& ||\mathbf{x^{(0)}} - \mathbf{x}^*||^2 - 2\sum_{i=0}^{k-1} \alpha^{(i)}(f(\mathbf{x^{(i)}}) - f(\mathbf{x}^*)) + \sum_{i=0}^{k-1}(\alpha^{(i)})^2||\mathbf{g^{(i)}}||^2 && (7) \end{aligned}$$

By assuming $||\mathbf{x^{(0)}} - \mathbf{x}^*||^2$ is some finite, bounded value R^2, and bounding our chosen subgradients by G, we have

$$0 \leq R^2 - 2\sum_{i=0}^{k-1} \alpha^{(i)}(f(\mathbf{x^{(i)}}) - f(\mathbf{x}^*)) + \sum_{i=0}^{k-1}(\alpha^{(i)})^2||\mathbf{g^{(i)}}||^2 \tag{8}$$

$$\iff (f(\mathbf{x^{(i)}}) - f(\mathbf{x}^*)) \leq \frac{R^2 + G^2 \sum_{i=0}^{k-1}(\alpha^{(i)})^2}{2\sum_{i=0}^{k-1} \alpha^{(i)}}.$$

Certainly, our best objective so far f_{best} satisfies

$$(f_{\text{best}} - f(\mathbf{x}^*)) \leq \frac{R^2 + G^2 \sum_{i=0}^{k-1} (\alpha^{(i)})^2}{2 \sum_{i=0}^{k-1} \alpha^{(i)}}.$$

This inequality is critical to help us obtain convergence guarantees by our chosen step sizes. Suppose we choose the constant step size ($\alpha^{(k)} = \alpha > 0 \in \mathbf{R}$), then

$$(f_{\text{best}} - f(\mathbf{x}^*)) \leq \frac{R^2 + G^2 k \alpha^2}{2k\alpha} \xrightarrow{k \to \infty} \frac{G^2 \alpha}{2}.$$

So choosing a constant step size allows us to converge eventually to within $\frac{G^2 \alpha}{2}$ of the optimal objective. We also observe that to obtain convergence, we want to choose step sizes to minimize $\sum_{i=0}^{k-1} (\alpha^{(i)})^2$ while maximizing $\sum_{i=0}^{k-1} \alpha^{(i)}$. One such candidate is $a^{(k)} := \frac{1}{k}$ since it is well known by the p-series test that the infinite series $\sum_{i=0}^{\infty} \frac{1}{n^p}$ converges for $p > 1$ and diverges for $p = 1$. By this choice of $a^{(k)}$, we have

$$(f_{\text{best}} - f(\mathbf{x}^*)) \leq \frac{R^2 + G^2 \sum_{i=0}^{k-1} \frac{1}{n^2}}{2 \sum_{i=0}^{k-1} \frac{1}{n}} \xrightarrow{k \to \infty} 0,$$

which completely converges! Using similar techniques, 5 types of steps are popular:

1. Constant $\alpha^{(k)} = \alpha > 0$
2. Constant normalized $\alpha^{(k)} = \gamma / ||g^{(k)}||_2$
3. Square summable but not summable $\sum_{i=k}^{\infty} \alpha^{(k)} = \infty, \sum_{i=k}^{\infty} (\alpha^{(k)})^2 < \infty$
4. Diminishing but not summable $\sum_{i=k}^{\infty} \alpha^{(k)} = \infty, \lim_{k \to \infty} \alpha^{(k)} = 0$
5. Diminishing normalized, but not summable $\alpha^{(k)} = \gamma^{(k)} / ||g^{(k)}||_2, \sum_{i=k}^{\infty} \gamma^{(k)} = \infty, \lim_{k \to \infty} \gamma^{(k)} = 0.$

2.3 Polyak's Step Length

Polyak's Step Length is a choice of step length when the optimal value f^* is known. The intuition is to minimize upperbound (5) between the kth step and an optimal value, which yields.

$$\alpha_k = \frac{f(x^{(k)}) - f^*}{||g^{(k)}||_2^2}. \tag{9}$$

To demonstrate convergence, we plug (9) into (8) to get

$$2 \sum_{i=0}^{k-1} \alpha^{(i)} (f(\mathbf{x^{(i)}}) - f(\mathbf{x}^*)) - \sum_{i=0}^{k-1} (\alpha^{(i)})^2 ||\mathbf{g^{(i)}}||^2 \leq R^2$$

$$\iff \sum_{i=0}^{i-1} \frac{f(\mathbf{x^{(i)}}) - f^*}{||g^{(i)}||_2^2} \leq R^2$$

Once again using $||g^{(k)}||_2^2 \leq G^2$ gives

$$\sum_{i=0}^{k-1} f(\mathbf{x^{(i)}}) - f^* \leq R^2 G^2.$$

Thus, we must have $f(x^{(k)}) \to f^*$. Otherwise, suppose $f(x^{(k)}) - f^* = r > 0, \forall k$, then

$$\lim_{k \to \infty} \sum_{i=0}^{k-1} f(x^{(k)}) - f^* \leq \lim_{k \to \infty} \sum_{i=0}^{k-1} r = \infty \not\leq R^2 G^2 \text{ (contradiction)}$$

so we have convergence. When f^* is not known, we can estimate it with $f_{best} - \gamma_i$ for $\gamma_i > 0, \gamma_i \to 0$ but $\sum_{i=0}^{\infty} \gamma_i = \infty$. This gives the step-size

$$\alpha_k = \frac{f(x^{(k)}) - (f_{best} - \gamma_k)}{||g^{(k)}||_2^2}.$$

We similarly substitute to demonstrate convergence.

$$\begin{aligned}
R^2 &\geq 2\sum_{i=0}^{k-1}\alpha^{(i)}(f(\mathbf{x^{(i)}})-f(\mathbf{x}^*))-\sum_{i=0}^{k-1}(\alpha^{(i)})^2||\mathbf{g^{(i)}}||^2 \\
&=\sum_{i=0}^{k-1}\frac{2(f(\mathbf{x^{(i)}})-(f_{best}-\gamma_i))(f(\mathbf{x^{(i)}})-f(\mathbf{x}^*))}{||g^{(i)}||_2^2}-\frac{(f(\mathbf{x^{(i)}})-(f_{best}-\gamma_i))^2}{||g^{(i)}||_2^2} \\
&=\sum_{i=0}^{k-1}\frac{(f(\mathbf{x^{(i)}})-f_{best}+\gamma_i)(f(\mathbf{x^{(i)}})+f_{best}-\gamma_i-2f(\mathbf{x}^*))}{||g^{(i)}||_2^2} \\
&\geq\sum_{i=0}^{k-1}\frac{(f(\mathbf{x^{(i)}})-f_{best}+\gamma_i)(2(f(\mathbf{x^{(i)}})-f(\mathbf{x}^*))-\gamma_i)}{||g^{(i)}||_2^2}.
\end{aligned}$$

The key now is to observe that as long as $f(\mathbf{x^{(i)}})-f(\mathbf{x}^*)=d>0$, there is always some iteration N at which $\gamma_i<d, \forall i\geq N$. Then $2(f(\mathbf{x^{(i)}})-f(\mathbf{x}^*))-\gamma_i)=2d-\gamma_i\geq d$. We extract the finite sum up to N and observe

$$\begin{aligned}
R^2-&\sum_{i=0}^{N}\frac{(f(\mathbf{x^{(i)}})-f_{best}+\gamma_i)(2(f(\mathbf{x^{(i)}})-f(\mathbf{x}^*))-\gamma_i)}{||g^{(i)}||_2^2} \\
&\geq\sum_{i=N+1}^{k-1}\frac{(f(\mathbf{x^{(i)}})-f_{best}+\gamma_i)(2(f(\mathbf{x^{(i)}})-f(\mathbf{x}^*))-\gamma_i)}{||g^{(i)}||_2^2} \\
&\geq\sum_{i=N+1}^{k-1}\frac{(f(\mathbf{x^{(i)}})-f_{best})}{G^2}+\frac{d\gamma_i}{G^2}
\end{aligned}$$

since we forced the infinite sequence $\{\gamma_i\}$ to converge to infinity, the right side converges to infinity. However, it is finitely bounded by the left side. Thus, k cannot be arbitrarily large.

The rest of the report discusses extensions of the subgradient method at a high level and showcases the methods for the general assignment problem.

3 Extended Methods

3.1 Projected Subgradient Descent

This algorithm is an extension of the basic subgradient descent algorithm. It address specifically problems of form

$$\begin{aligned}
&\min_{x\in C} f(x) \\
&\text{s.t. } x\in C
\end{aligned}$$

and works by projecting each iteration back into C after stepping.

3.2 Incremental Subgradient Descent [3]

One extension to the basic subgradient method is an approach by Nedic and Bertsekas – the Incremental Subgradient Descent method. This group investigates specifically a primal separable combinatorial problem of form

$$\begin{aligned}
&\max\sum_{i=1}^{m}c_i^T y_i \\
&\text{subject to } y_i\in Y_i, i=1,...,m, \sum_{i=1}^{m}A_i y_i\leq b,
\end{aligned}$$

where $c_i\in\mathbf{R^n}$ are given vectors, Y_i are finite subsets of $\mathbf{R^n}$, $A_i\in\mathbf{R^{n\times p}}$, and $b\in\mathbf{R^n}$. The dual problem is

$$\begin{aligned}
&\min f(x)=\sum_{i=1}^{m}f_i(x) \\
&\text{subject to } x\in\{x\in\mathbf{R^n}|x\geq 0\}
\end{aligned}$$

where

$$f_i(x) = \max_{y_i \in Y_i} (c_i + A_i^T x)^T y_i - \beta_i^T x, i = 1, ..., m$$

$$\beta_i \in \mathbf{R^n}, \sum_{i=1}^{m} \beta_i = b.$$

and x is the Lagrange multiplier vector. One method for solving this problem is the projected subgradient method.

$$x_{k+1} = \text{proj}_X \left[x_k - \alpha_k \sum_{i=1}^{m} d_{i,k} \right]$$

where $d_{i,k}$ is a subgradient of f_i at x_k. Instead, the incremental subgradient method suggests taking one incremental step for each f_i at each iteration.

$$x_{k+1} = \phi_{m,k}$$

where

$$\phi_{i,k} = \text{proj}_X \left[\phi_{i-1,k} - \alpha_k g_{i,k} \right], g_{i,k} \in \delta f_i(\phi_{i-1,k}).$$

3.3 Modified Gradient [4]

This method modifies the subgradient to be

$$s^{(k)} = g^{(k)} + \beta_k s^{(k-1)}$$

where $g^{(k)}$ is the subgradient at $x^{(k)}$ and β_m is a suitable constant. Essentially, this variation adds a weighted combination of previous gradients into the current gradient. I do not outline the convergence proofs and techniques here since there is too much to show but instead refer to [4]. The researchers were able to show that using their weighted subgradient scheme, they can guarantee results at least as good as the basic subgradient method.

3.4 The Generalized Assignment Problem [3]

We implement the incremental method, and compare it to the regular projected subgradient method on different choices of step-size. We also implement the heavy ball method. The code can be found here.

The problem is to assign m jobs to n machines. Machine j performing job i costs a_{ij} and requires p_{ij} time units. We want to

$$\begin{aligned} \text{minimize } & \sum_{i=1}^{m} \sum_{j=1}^{n} a_{ij} y_{ij} \\ \text{subject to } & \sum_{j=1}^{n} y_{ij} = 1, i = 1, ..., m \\ & \sum_{i=1}^{m} p_{ij} y_{ij} \leq t_j, j = 1, ..., n \\ & y_{ij} \in \{0, 1\}, \forall i, j \end{aligned}$$

The dual problem is

$$\begin{aligned} \max f(x) &= \sum_{i=1}^{m} f_i(x) \\ \text{subject to } & x \geq 0, \end{aligned}$$

where

$$f_i(x) = \min_{y_{ij}=0 \text{ or } 1, \sum_{j=1}^{n} y_{ij} = 1} \sum_{j=1}^{n} (a_{ij} + x_j p_{ij}) y_{ij} - \frac{1}{m} \sum_{j=1}^{m} t_j x_j, \ i = 1, ..., m$$

A subgradient of f_i at x is given by

$$g = (g_1, ..., g_n)^T, \ g_j = \begin{cases} -\frac{t_j}{m} & \text{if } j \neq j^* \\ p_{ij^*} - \frac{t_j}{m} & \text{if } j = j^* \end{cases}$$

where j^* is the minimizer of

$$a_{ij} + x_j p_{ij}$$

since y_{ij} is true for only one j for each i.

3.5 Discussion (Implementation Reflection / What I've learned)

Many variables could be tuned in the task assignment problem. I've spent a long time debugging my code to make sure everything should be implemented correctly and tuning without being able to get consistent results for every problem configuration (when new matrices are generated, the variables need to be re-tuned), so I've concluded that tuning the methods for specific questions is some we inherently have to do. Here are some challenges I've faced during tuning. For my problem, I tried using $N = 4$, $M = 100$. I also set the boundaries for T in such a way that the problem is always feasible, following [3].

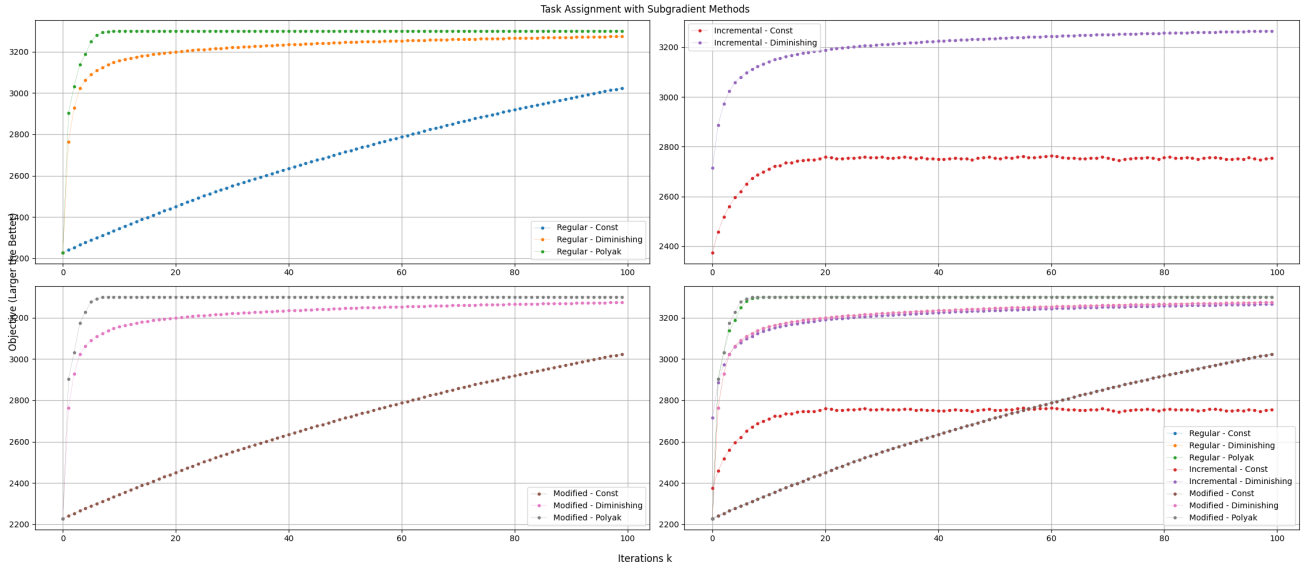
First, if the problem $f(x)$ we are solving isn't smooth enough and is very sensitive, even if we approach an optimizer, we could have no clue since the objective could vary wildly near the optimum. Further, the lack of a good stopping criterion makes it difficult to judge when to stop, or when you would get there, even though in theory you will "get there" over enough iterations. We can see in the convergence proofs how $||x^{(k)} - x^*||$ is being bounded, and not $||f(x^{(k)}) - f(x^*)||$.

Next, problem sensitivity is a really big issue that's closely tied to choosing good step sizes. For step size in the form of sequences, choosing the convergence rate is a non trivial task too. The first problem is that you might not have a great idea of what the optimal value of this problem is, and the method could converge so slowly that you wouldn't know if the objectives you've computed are close to it. You are merely promised "eventually convergence". So sometimes Polyak steps are not possible. The technique to estimate Polyak steps assumes that f_{best} converges to f^* at some rate which we model as a sequence that converges to 0 but sums to infinity. However, specifically for my problem, I believe convergence is slow, which makes Polyak step estimation unsuccessful.

In terms of reliability, constant steps are consistent, but a good step needs to be empirically found for each problem. And eventual convergence is not guaranteed. Diminishing steps such as $\frac{1}{n^k}$ which guaranteed convergence seemed promising but was not always effective. The problem was mainly that the tradeoffs of using a fast-converging diminishing step which the convergence proofs suggest. Suppose the first few "bigger steps" are really bad and send the objective in the opposite direction, the convergence rate would actually work against the algorithm since it would have to spend a long time recovering from these bad steps using smaller and smaller steps. It's also difficult to know if you've converged to a value or if you're simply just taking tiny steps since the steps converged very quickly. Having backtracking (jump back to best x if the most recent 5 steps have been bad) helped with this issue.

I also believe numerical accuracy made tuning more tricky as well. This could relate to the sensitivity of my problem, and I believe it would get worse as the problem scales.

Overall, in my testing, it was tricky to work with subgradient methods even though the theory seems promising. There's a lot to tune and change and there's information that can be critical for good convergence rates (like optimal value) that we might not always have. But these methods enable us to do convex optimization in non-differentiable settings and achieve convergence guarantees, which is already very good. Overall, for this specific problem, I didn't observe a huge difference across methods and noticed that the choice of step size is a bigger factor for convergence rate. Below is a plot of some of my results. You can see how constant step sizes converges slows, and for the incremental method the constant step size did not acheive the optimal solution (as it is not guaranteed). The Polyak method was most effective across the different methods.



In terms of directions to explore more, I'm curious whether there's a way to estimate the Hessian in non-differential settings as well so that we can maybe attain a second order subgradient method for faster convergence. Getting the subgradient is also very problem specific so I wonder if there was a more systematic or algorithmic way of obtaining it. I also wanted to explore stochastic subgradient methods and add randomness into the methods but was limited

on time.

References

[1] S. Boyd, L. Xiao, and A. Mutapcic, "Subgradient methods," *lecture notes of EE392o, Stanford University, Autumn Quarter*, vol. 2004, no. 01, 2003.

[2] D. Bertsekas, *Convex optimization theory*, vol. 1. Athena Scientific, 2009.

[3] A. Nedic and D. P. Bertsekas, "Incremental subgradient methods for nondifferentiable optimization," *SIAM Journal on Optimization*, vol. 12, no. 1, pp. 109–138, 2001.

[4] P. M. Camerini, L. Fratta, and F. Maffioli, *On improving relaxation methods by modified gradient techniques*, pp. 26–34. Berlin, Heidelberg: Springer Berlin Heidelberg, 1975.