Large Scale Optimization for Machine Learning:
Lecture 21

Lecturer: Meisam Razaviyayn Scribe: Nitin Kamra

Nov 7-9, 2017

1 Multi-Block Structure and BCD Method

In most cases in ML, we need to solve a problem of the following form:

$$\min_{\mathbf{w}} \frac{1}{n} \sum_{i=1}^{n} l((\mathbf{x}_i, y_i), \mathbf{w}) + \lambda \mathcal{R}(\mathbf{w})$$

When the regularizer term $\mathcal{R}(\mathbf{w})$ is convex but non-smooth (e.g., l_1 norm), there are many methods that can be used. From previous lectures, we know that the Proximal Gradient method can exploit the structure with the non-smoothness of the regularizer and the Incremental (Stochastic) Proximal Gradient method can deal with the case of summation structure. Proximal Gradient method is generally fast assuming that each iteration is easy to compute. Sub-gradient descent is also feasible, but is typically slow and has no good termination criteria other than cross validation.

1.1 Multi-Block Structure

Now, we will show how to exploit the above structure when variables are distributed in multiple blocks i.e. we have the following structure:

$$\begin{aligned} \min_{\mathbf{x}} \quad & f(\mathbf{x}_1, \ldots, \mathbf{x}_m) \\ \text{s.t.} \quad & \mathbf{x}_i \in \mathcal{X}_i,\ i = 1, \ldots, m \end{aligned}$$

1.2 Block Coordinate Descent (BCD)

To optimize problems having the multi-block structure, we use the *Block Coordinate Descent (BCD)* method (Algorithm 1).

Algorithm 1 Sketch of the BCD Method

At iteration r, choose an index i and update

1: $\mathbf{x}_i^{r+1} = \arg\min_{\mathbf{x}_i \in \mathcal{X}_i} f(\mathbf{x}_1^r, \dots \mathbf{x}_{i-1}^r, \mathbf{x}_i, \mathbf{x}_{i+1}^r, \dots, \mathbf{x}_m^r)$

2: $\mathbf{x}_k^{r+1} = \mathbf{x}_k^r, \quad \forall k \neq i$

Note that Algorithm 1 leaves us the freedom of how to choose the index i at each iteration. There are several popular rules: (a) randomized i.e., uniformly randomly choose an i at each iteration, (b) cyclic i.e., iterate over these m different $\mathbf{x}_i$'s, (c) greedy i.e., evaluate the new objective value by making update to each $\mathbf{x}_i$ separately and choose the best one, and (d) parallel i.e., parallelly update every i at each iteration. It is worth noting that the greedy selection rule could be costly with large number of variables. Also, we should be careful when using parallel update rule. In particular, it could happen that updating either $\mathbf{x}_i$ or $\mathbf{x}_j$ will decrease the objective, however updating both $\mathbf{x}_i, \mathbf{x}_j$ will cause an increase of the objective (see figure 1).

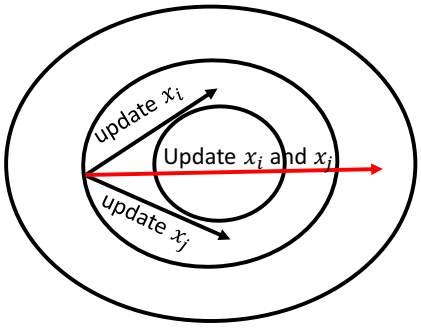

Figure 1: Parallel updates in BCD

1.3 Example

An application of BCD method is the r-rank matrix approximation problem. We know the rank-1 matrix approximation problem can be described as:

$$\min_{\mathbf{x},\mathbf{y}} \|\mathbf{A} - \mathbf{x}\mathbf{y}^T\|_F^2$$

The rank-r approximation problem can similarly be described as:

$$\begin{aligned} \min_{\mathbf{Z}} \quad & \|\mathbf{A} - \mathbf{Z}\|_F^2 \\ \text{s.t.} \quad & \text{rank}(\mathbf{Z}) \leq r \end{aligned}$$

We can approximate this problem by substituting the rank constraint with a nuclear norm-based relaxation:

$$\min_{\mathbf{Z}} \|\mathbf{A} - \mathbf{Z}\|_F^2 + \lambda \|\mathbf{Z}\|_*$$

But when the dimensions of matrix $\mathbf{A}$ (and hence of $\mathbf{Z}$) are very large, solving the problem becomes computationally very expensive. In such cases, we can solve the original problem by using BCD method.

First, note that a matrix has rank at most r iff it can be decomposed as $\mathbf{X}\mathbf{Y}^T$, where $\mathbf{X}$ is a matrix with dimensions $n \times r$ and $\mathbf{Y}$ is a matrix with dimensions $m \times r$. So, the above problem can be reformulated as:

$$\min_{\mathbf{X},\mathbf{Y}} \|\mathbf{A} - \mathbf{X}\mathbf{Y}^T\|_F^2,$$

which is a non-convex problem. But, if we fix $\mathbf{X}$ (or $\mathbf{Y}$) and solve for $\mathbf{Y}$ (resp. $\mathbf{X}$), then it becomes a convex problem. For example, fixing $\mathbf{Y}$ and solving for $\mathbf{X}$, we will have:

$$(\mathbf{X}\mathbf{Y}^T - \mathbf{A})\mathbf{Y} = \mathbf{0} \implies \mathbf{X} = \mathbf{A}\mathbf{Y}(\mathbf{Y}^T\mathbf{Y})^{-1}$$

Here, we just need to compute the inverse of a matrix with dimension $r \times r$, which is easier since r is smaller. So, the BCD method provides an easier way to solve this problem.

2 Convergence of the BCD Method

It turns out that the BCD method will converge to stationary points under proper assumptions. The following proposition [1] gives the convergence of BCD method:

Proposition 1. *Suppose that f is continuously differentiable over the set X. Furthermore, suppose that for each $\mathbf{x} = (\mathbf{x}_1, \ldots, \mathbf{x}_m) \in X$ and i,*

$$f(\mathbf{x}_1^r, \ldots, \mathbf{x}_{i-1}^r, \eta, \mathbf{x}_{i+1}^r, \ldots, \mathbf{x}_m^r)$$

viewed as a function of η, attains a unique minimum $\bar{\eta}$ over X_i, and is monotonically nonincreasing along the line from $\mathbf{x}_i$ to $\hat{\eta}$. Let $\{\mathbf{x}^k\}$ be the sequence generated by the block coordinate descent method. Then every limit point of $\{\mathbf{x}^k\}$ is a stationary point.

Proof. We use the cyclic rule, and let $\mathbf{z}_i^k = (x_1^{k+1}, \cdots, x_i^{k+1}, x_{i+1}^k, \cdots, x_m^k)$. From the algorithm, we have:

$$f(\mathbf{x}^k) \geq f(\mathbf{z}_1^k) \geq f(\mathbf{z}_2^k) \geq \ldots \geq f(\mathbf{z}_{m-1}^k) \geq f(\mathbf{x}^{k+1})) \quad \forall k. \tag{1}$$

Define $\bar{\mathbf{x}} \triangleq (\bar{x}_1, \ldots, \bar{x}_m)$ to be a limiting point of the sequence $\{\mathbf{x}^k\}$ ($\bar{\mathbf{x}} \in X$ as X is closed). From equation 1 and the definition of $\bar{\mathbf{x}}$, we know that the whole sequence $\{f(\mathbf{x}^k)\}$ converges to $f(\bar{\mathbf{x}})$. Let $\mathbf{x}^{k_j}, j = 1, 2, \ldots$ be the subsequence of $\mathbf{x}^k$ such that it converges to $\bar{\mathbf{x}}$. We first show that $\{\mathbf{x}_1^{k_j+1} - \mathbf{x}_1^{k_j}\}$ converges to zero as $j \longrightarrow \infty$.

First assume otherwise, then it is equivalent to say that $\{\mathbf{z}_1^{k_j} - \mathbf{x}^{k_j}\}$ does not converge to zero. Let $\gamma^{k_j} = \|\mathbf{z}_1^{k_j} - \mathbf{x}^{k_j}\|$. By further restricting to a subsequence of $\{k_j\}$ say $\{k_t'\}$, we can have that $\exists\, \bar{\gamma}$ such that $\gamma^{k_t'} \geq \bar{\gamma}\ \forall t$. Define $\mathbf{s}_1^{k_t'} = \frac{\mathbf{z}_1^{k_t'} - \mathbf{x}^{k_t'}}{\gamma^{k_t'}}$, so $\mathbf{z}_1^{k_t'} = \mathbf{x}^{k_t'} + \gamma^{k_t'} \mathbf{s}_1^{k_t'}$ and $\|\mathbf{s}_1^{k_t'}\| = 1$. Hence, $\mathbf{s}_1^{k_t'}$ belongs to a compact set and thus has a limiting point, say $\bar{\mathbf{s}}_1$. Without loss of generality, we assume that the whole sequence of $\mathbf{s}_1^{k_t'}$ converges to a point $\bar{\mathbf{s}}_1$.
If we fix some $\epsilon \in [0, 1]$, so $0 \leq \epsilon\bar{\gamma} \leq \gamma^{k_t'}$, which means $\mathbf{x}^{k_t'} + \epsilon\bar{\gamma}\mathbf{s}_1^{k_t'}$ lies on the segment joining $\mathbf{x}^{k_t'}$ and $\mathbf{x}^{k_t'} + \gamma^{k_t'}\mathbf{s}_1^{k_t'} = \mathbf{z}_1^{k_t'}$, and belongs to X because X is a convex set.

Based on above definitions, we have: $f(\mathbf{z}_1^{k_t'}) = f(\mathbf{x}^{k_t'} + \gamma^{k_t'}\mathbf{s}_1^{k_t'}) \leq f(\mathbf{x}^{k_t'} + \epsilon\bar{\gamma}\mathbf{s}_1^{k_t'}) \leq f(\mathbf{x}^{k_t'})$. Because from equation 1, we know $f(\mathbf{z}_1^{k_t'}) \longrightarrow f(\bar{\mathbf{x}})$, so if

we take $t \longrightarrow \infty$, then we have, $f(\bar{\mathbf{x}} + \epsilon\bar{\gamma}\bar{\mathbf{s}_1}) = f(\bar{\mathbf{x}})$, for every $\epsilon \in [0, 1]$. Since $\bar{\gamma}\bar{\mathbf{s}_1} \neq \mathbf{0}$, this contradicts the assumption that f is uniquely minimized for block variables [1]. So, the above analysis establishes that $x_1^{k_j+1} - x_1^{k_j}$ converges to zero, which means $\mathbf{z}_1^{k_j} \longrightarrow \bar{\mathbf{x}}$.

Further, based on the definitions, we have: $f(\mathbf{z}_1^{k_j}) \leq f(x_1, x_2^{k_j}, \ldots, x_m^{k_j}), \forall x_1 \in X_1$. Taking the limit as j tends to infinity, we have: $f(\bar{\mathbf{x}}) \leq f(x_1, \bar{x}_2, \ldots, \bar{x}_m), \forall x_1 \in X_1$. So, from the first order condition for a convex problem, we have: $\nabla_1 f(\bar{\mathbf{x}})'(x_1 - \bar{x}_1) \geq 0, \forall x_1 \in X_1$.
With similar analysis for each block variable, we have $\nabla_i f(\bar{\mathbf{x}})'(x_i - \bar{x}_i) \geq 0$, $\forall x_i \in X_i$ for any i. Adding these inequalities and using the Cartesian product structure of the set X, we conclude that $\nabla f(\bar{\mathbf{x}})'(\mathbf{x} - \bar{\mathbf{x}}) \geq \mathbf{0}, \forall \mathbf{x} \in X$. □

3 Necessity of assumptions for convergence

There are three key assumptions in Proposition 1: (a) separable constraints, (b) differentiable objective, and (c) unique minimizer at each step. We will now argue that each of these assumptions is necessary, though we will also see that the assumption of differentiable objective could be slightly relaxed.

3.1 Necessity of Separable Constraints

Consider the problem of minimizing $x^2 + y^2$ subject to $x + y = 0$. The BCD method will get stuck at the initial point. For example, if we start with $x = 1$ and $y = -1$, then at each iteration, updating either x or y will not change the its value at all due the the equality constraint. Therefore, having separable constraints on variables is necessary.

[1] Because $f(\mathbf{z}_1^{k_j}) \leq f(x_1, x_{-1}^{k_j})\ \forall x_1 \in X_1$, taking limit of j, we have that $f(\bar{\mathbf{x}}) = f(\bar{x}_1, \bar{x}_{-1}) \leq f(x_1, \bar{x}_{-1})\ \forall x_1 \in X_1$. Then given $\bar{x}_{-1}$, $\bar{x}_1$ should be the unique minimizer of $f(x_1, \bar{x}_{-1})$ which contradicts with the equality $f(\bar{\mathbf{x}} + \epsilon\bar{\gamma}\bar{\mathbf{s}_1}) = f(\bar{\mathbf{x}})$.

3.2 Necessity of Smooth Objective and a Slight Generalization

Consider the example of minimizing $\|\mathbf{Ax}\|_1$ where $\mathbf{A} = [3, 4; 2, 1]$ (see figure 2 for its contour plot). We can see that at point $(-4, 3)$, updating either x_1

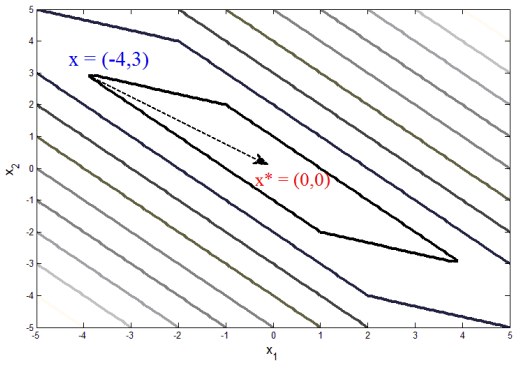

Figure 2: Contour plot of $\|\mathbf{Ax}\|_1$ where $\mathbf{A} = [3, 4; 2, 1]$

or x_2 will not cause an improvement of the objective value. However, $(-4, 3)$ is not a stationary point since the function value strictly increases along the direction $(4, -3)$. This shows that the assumption of smooth objectives is necessary.

In the following, we discuss a slightly relaxed assumption of the smoothness which still leads to the convergence. We first introduce the concept of *regular* functions.

Definition 1. *A function $f(\cdot)$ is* regular *at point $\mathbf{z}$, if for any direction $\mathbf{d} = (\mathbf{d}_1, \ldots, \mathbf{d}_k, \ldots, \mathbf{d}_m)$* [1]*,*

$$f'(\mathbf{z}; (0, \ldots, 0, \mathbf{d}_k, 0, \ldots, 0)) \geq 0, \forall \mathbf{d}_k, \forall k \implies f'(\mathbf{z}; \mathbf{d}) \geq 0$$

Remarks:

- When $f(\cdot)$ is restricted to a region X. Then f is regular in region X if the above condition is true for any feasible direction d_k and $\mathbf{d}$ in X.
- Any differentiable function is regular since $f'(\mathbf{z}; (0, \ldots, 0, \mathbf{d}_k, 0, \ldots, 0)) = \langle \mathbf{d}_k, \nabla_k f(\mathbf{z}) \rangle \geq 0$ and $f'(\mathbf{z}; \mathbf{d}) = \langle \nabla f(\mathbf{z}), \mathbf{d} \rangle = \sum_k \langle \mathbf{d}_k, \nabla_k f(\mathbf{z}) \rangle \geq 0$.

[1] Here k is the block index, hence $\mathbf{d}_k$ is a vector.

The following is a useful tool that helps us to judge a function is regular.
Claim 1. *If $g(\mathbf{x})$ is differentiable, then $f(\mathbf{x}) = g(\mathbf{x}) + \sum_k h_k(x_k)$ is regular.*

Proof. When g is differentiable, $d_k \nabla_k f(\mathbf{x}) = |d_k| g'(\mathbf{x}; (0, \ldots, \frac{d_k}{|d_k|}, \ldots, 0))$. Therefore, we have

$$\begin{aligned} f'(\mathbf{x}; \mathbf{d}) &= g'(\mathbf{x}; \mathbf{d}) + \sum_k h_k'(x_k; \mathbf{d}) \\ &= \sum_k |d_k| g'(\mathbf{x}; (0, \ldots, \frac{d_k}{|d_k|}, \ldots, 0)) + \sum_k |d_k| h_k'(x_k; \frac{d_k}{|d_k|}) \\ &= \sum_k |d_k| \left[g'(\mathbf{x}; (0, \ldots, \frac{d_k}{|d_k|}, \ldots, 0)) + h_k'(x_k; \frac{d_k}{|d_k|}) \right] \\ &= \sum_k |d_k| \cdot f'(\mathbf{x}; (0, \ldots, \frac{d_k}{|d_k|}, \ldots, 0)) \end{aligned}$$

Therefore, $f'(\mathbf{x}; \mathbf{d}) \geq 0$ if $f'(\mathbf{x}; (0, \ldots, \frac{d_k}{|d_k|}, \ldots, 0)) \geq 0$ for each k. □

One example of Claim 1 is to minimize $||Ax - b||_2^2 + \lambda ||x||_1$. It turns out that relaxing the smoothness assumption in Proposition 1 to the regularity condition defined above also yields convergence. In particular,
Theorem 1 (Tseng [2])**.** *Suppose that f is regular over the set X and X is compact. Furthermore, suppose that for each $\mathbf{x} = (\mathbf{x}_1, \ldots, \mathbf{x}_m) \in X$ and i,*

$$f(\mathbf{x}_1^r, \ldots \mathbf{x}_{i-1}^r, \eta, \mathbf{x}_{i+1}^r \ldots, \mathbf{x}_m^r)$$

viewed as a function of η, attains a unique minimum $\bar{\eta}$ over X_i, and is monotonically nonincreasing along the line from $\mathbf{x}_i$ to $\hat{\eta}$. Let $\{\mathbf{x}^k\}$ be the sequence generated by the block coordinate descent method. Then every limit point of $\{\mathbf{x}^k\}$ is a stationary point.

Recall that a point $\bar{\mathbf{x}}$ is *stationary* if $f'(\bar{\mathbf{x}}; \mathbf{x} - \bar{\mathbf{x}}) \geq 0, \forall \mathbf{x} \in X$. When f is smooth, the definition is equivalent to $\langle \nabla f(\bar{\mathbf{x}}), \mathbf{x} - \bar{\mathbf{x}} \rangle \geq 0, \forall \mathbf{x} \in X$. If further we have $X = \mathbb{R}^m$, the definition is equivalent to $\nabla f(\bar{x}) = 0$. Also, the above theorem is true for cyclic, randomized and greedy selection rules.

Remark: The rate of convergence of the BCD method for smooth optimization is similar to that of gradient descent: sublinear for general convex and

linear for strongly convex. Same convergence results can be shown for most of the popular non-smooth objectives.

3.3 Necessity of Unique Minimizer

In this subsection, we argue via an example, that the assumption of a unique minimizer is also necessary. The example is from Michael J. D. Powell. Define

$$(x-c)_+^2 = \begin{cases} 0, & \text{if } x \le c \\ (x-c)^2, & \text{if } x \ge c \end{cases}$$

Let $f(x,y,z) = -xy - yz - xz + (x-1)_+^2 + (-x-1)_+^2 + (y-1)_+^2 + (-y-1)_+^2 + (z-1)_+^2 + (-z-1)_+^2$ for $x, y, z \in \mathbb{R}$. Notice that f is first-order differentiable because $(x-c)_+^2$ is. Let (x^r, y^r, z^r) be the point at iteration r. Without loss of generality, assume that we want to update x. It is easy to check the BCD method will update x^{r+1} as follows

$$x^{r+1} = \begin{cases} 1 + \frac{y^r+z^r}{2} & \frac{y^r+z^r}{2} > 0 \\ \text{any value in } [-1,1] & \frac{y^r+z^r}{2} = 0 \\ -1 + \frac{y^r+z^r}{2} & \frac{y^r+z^r}{2} < 0 \end{cases}$$

Notice that when $\frac{y^r+z^r}{2} = 0$, the minimizer of f as a function of x is not unique. It turns out that this causes issues. In particular, if we start at the point $(-1-\epsilon, 1+\epsilon/2, -1-\epsilon/4)$, updating x we get $(1+\epsilon/8, 1+\epsilon/2, -1-\epsilon/4))$ since $y^r + z^r > 0$ in this case and $x^{r+1} = 1 + \frac{y^r+z^r}{2} = 1 + \epsilon/8$. After similar updates as before, we get point sequence $(-1-\epsilon, 1+\epsilon/2, -1-\epsilon/4) \to (1+\epsilon/8, 1+\epsilon/2, -1-\epsilon/4) \to (1+\epsilon/8, -1-\epsilon/16, -1-\epsilon/4) \to (1+\epsilon/8, -1-\epsilon/16, 1+\epsilon/32) \to (-1-\epsilon/64, -1-\epsilon/16, 1+\epsilon/32) \to (-1-\epsilon/64, 1+\epsilon/128, 1+\epsilon/32) \to (-1-\epsilon/64, 1+\epsilon/128, -1-\epsilon/256) \to \ldots$. A limit point of this sequence will be a point of entries 1 or -1, but not all 1 or all -1. It is easy to check that neither of these are stationary points.

Though the example above is specifically crafted and happens very rarely, the problem of non-unique minimizer does happen in practical problems, e.g., tensor decomposition. A tensor is a three-dimensional analogue of matrix, i.e., a block of real numbers. Given a tensor $\mathcal{X} \in \mathbb{R}^{I \times J \times K}$, the tensor PARAFAC decomposition problem is to find $A \in \mathbb{R}^{I \times L}, B \in \mathbb{R}^{J \times L}, C \in$

$\mathbb{R}^{K\times L}$, such that $\mathcal{X} = \sum_{l\in L} \mathbf{a}_l \circ \mathbf{b}_l \circ \mathbf{c}_l$. L is called the rank of the tensor. Finding the tensor rank is NP-hard [3]. The decomposition can be formulated as the following optimization problem:

$$\min_{A,B,C} \left\| \mathcal{X} - \sum_{l\in L} \mathbf{a}_l \circ \mathbf{b}_l \circ \mathbf{c}_l \right\|.$$

The alternating least square algorithm [Carroll 1970, Harshman1970] is an example of the BCD method. In practice, we observe the *swamp* effect and slow convergence due to non-uniqueness of minimizer (see figure 3).

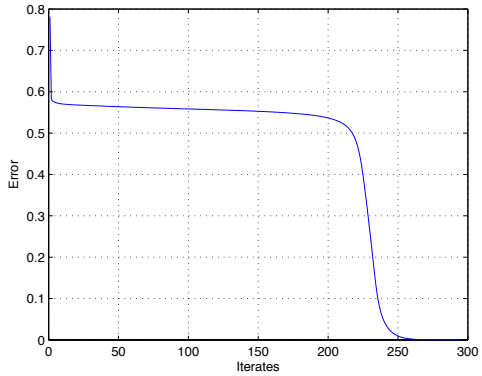

Figure 3: Swamp effect in Tensor PARAFAC decomposition

3.4 Limitations of BCD method

In summary, the BCD method suffers from the following limitations: (a) Requires separable constraints, (b) relies on uniqueness of the minimizer, and (c) requires regularity assumption.

Another major drawback of BCD algorithms is that the algorithm needs to compute an exact minimizer for the sub-problem at each iteration. The algorithm is very expensive unless the sub-problem is easy to solve. One way to remedy this is to do inexact BCD, which generally involves replacing $f(x_i, x^r_{-i})$ by a convex upper bound $u_i(x_i, x^r_{-i})$ and then optimizing the upper bound.

4 Block Successive Upper-bound Minimization

As proposed in the last section, the idea behind Block Successive Upper-bound Minimization is to approximate $f(x_i, x^r_{-i})$ by a convex function $u_i(x_i, x^r_{-i})$ which over-estimates $f(x_i, x^r_{-i})$ (see figure 4). We require $u_i(\cdot)$ to possess the following properties:

- **Global upper bound**: $u_i(x_i, x^r_{-i}) \geq f(x_i, x^r_{-i})$ for any x_i.
- **Locally tight**: $u_i(\mathbf{x}^r) = f(\mathbf{x}^r)$ and $u'(x_i, x^r_{-i}; d_i)\Big|_{x_i = x^r_i} = f'(\mathbf{x}^r; \mathbf{d})$ for every $\mathbf{d} = (0, \ldots, d_i, \ldots, 0)$.

Here x_{-i} denote the vector $(x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_m)$. The function $u(x_i, x^r_{-i})$, is some local approximation of the objective function and is designed to be usually easy to solve.

We then update $x^{r+1}_i = \arg\min_{x_i} u(x_i, x^r_{-i})$. Notice that $f(x^{r+1}_i, x^r_{-i}) \leq u(x^{r+1}_i, x^r_{-i}) \leq u(x^r_i, x^r_{-i}) = f(x^r_i, x^r_{-i})$, therefore the update guarantees that the objective is non-increasing and hence the algorithm is monotone. Note that, we can design the upper bound u_i appropriately to have a unique minimizer for the upper-bounded objective, thereby allowing us to mitigate the unique minimizer assumption. Next, we will see some examples of this technique.

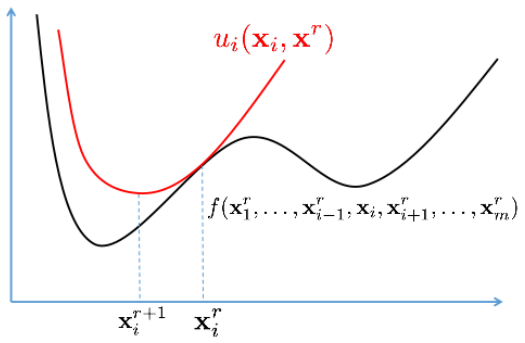

Figure 4: Block Successive Upper-bound Minimization

4.1 Block Coordinate (Proximal) Gradient Descent

4.1.1 Smooth scenario

For smooth objective functions, an upper bound can be constructed the same way as for gradient-descent:

$$u(x_i, \mathbf{x}^r) = f(\mathbf{x}^r) + \langle \nabla_{x_i^r} f(\mathbf{x}^r), x_i - x_i^r \rangle + \frac{1}{2\alpha} \|x_i - x_i^r\|_2^2$$

It can be verified using the Descent Lemma (and assuming $\alpha \leq \frac{1}{L}$) that this a global upper bound at $\mathbf{x}^r$, and is also locally tight. This results in a gradient descent update for the argmin step:

$$x_i^{r+1} = x_i^r - \alpha^r \nabla_{x_i^r} f(\mathbf{x}^r)$$

4.1.2 Non-smooth scenario

If the objective function $f = f_0 + f_1$ also involves a non-smooth term (f_0), we can define an upper bound similar to that of proximal gradient method:

$$u_i(x_i, \mathbf{x}^r) = f_0(\mathbf{x}^r) + f_1(\mathbf{x}^r) + \langle \nabla_{x_i^r} f_1(\mathbf{x}^r), x_i - x_i^r \rangle + \frac{1}{2\alpha} \|x_i - x_i^r\|_2^2$$

This results in a proximal gradient update for the argmin step:

$$x_i^{r+1} = x_i^r - \alpha^r \tilde{\nabla}_{x_i^r} f(\mathbf{x}^r; \alpha_r)$$

One can also use any Bregman divergence while creating the upper bound instead of $\frac{1}{2\alpha} \|x_i - x_i^r\|_2^2$.

4.1.3 Alternating Proximal Minimization

Another way of generating an upper bound is by adding a quadratic penalty term to the objective:

$$\mathbf{x}_i^{r+1} = \arg\min_{\mathbf{x}_i} \{ f(\mathbf{x}_1^r, \ldots, \mathbf{x}_{i-1}^r, \mathbf{x}_i, \ldots, \mathbf{x}_{i+1}^r, \ldots, \mathbf{x}_m^r) + \frac{\mu}{2} \|\mathbf{x}_i - \mathbf{x}_i^r\|_2^2 \}$$

It is easily verified that this a global upper bound and locally tight. This can sometimes make the objective convex and will always make an already convex function into a strongly convex one. Creating an upper bound this way, has been shown to mitigate the *swamp* effect in PARAFAC Tensor Decomposition.

4.2 Expectation Maximization Algorithm

The Expectation Maximization (EM) algorithm is a popular machine learning algorithm for parameter estimation in the presence of latent (unobserved) variables. Typically, the problem requires estimating parameters $\boldsymbol{\theta}$ for maximizing log-likelihood of observations $\mathbf{w}$ given some unobserved variables $\mathbf{z}$:

$$\hat{\boldsymbol{\theta}}_{ML} = \arg\max_{\boldsymbol{\theta}} \ln\, p(\mathbf{w}|\boldsymbol{\theta})$$

But directly minimizing the negative log-likelihood is hard and the problem is handled by first constructing a tight global upper bound $u(\boldsymbol{\theta}, \boldsymbol{\theta^r})$ for the objective at each iteration r and then optimizing the upper bound i.e.

$$\begin{aligned}
-\ln\, p(\mathbf{w};\boldsymbol{\theta}) &= -\ln\, \mathbb{E}_{\mathbf{z}|\boldsymbol{\theta}}\, p(\mathbf{w}|\mathbf{z},\boldsymbol{\theta}) \\
&= -\ln\, \mathbb{E}_{\mathbf{z}|\boldsymbol{\theta}} \left[\frac{p(\mathbf{z}|\mathbf{w},\boldsymbol{\theta}^r)p(\mathbf{w}|\mathbf{z},\boldsymbol{\theta})}{p(\mathbf{z}|\mathbf{w},\boldsymbol{\theta}^r)}\right] \\
&= -\ln\, \mathbb{E}_{\mathbf{z}|\mathbf{w},\boldsymbol{\theta}^r} \left[\frac{p(\mathbf{z}|\boldsymbol{\theta})p(\mathbf{w}|\mathbf{z},\boldsymbol{\theta})}{p(\mathbf{z}|\mathbf{w},\boldsymbol{\theta}^r)}\right] \\
&\leq -\mathbb{E}_{\mathbf{z}|\mathbf{w},\boldsymbol{\theta}^r} \ln \left[\frac{p(\mathbf{z}|\boldsymbol{\theta})p(\mathbf{w}|\mathbf{z},\boldsymbol{\theta})}{p(\mathbf{z}|\mathbf{w},\boldsymbol{\theta}^r)}\right] \qquad \text{(Jensen's inequality)} \\
&= -\mathbb{E}_{\mathbf{z}|\mathbf{w},\boldsymbol{\theta}^r} \ln\, p(\mathbf{w},\mathbf{z}|\boldsymbol{\theta}) + \mathbb{E}_{\mathbf{z}|\mathbf{w},\boldsymbol{\theta}^r} \ln\, p(\mathbf{z}|\mathbf{w},\boldsymbol{\theta}^r) \triangleq u(\boldsymbol{\theta},\boldsymbol{\theta}^r)
\end{aligned}$$

This results in a block-coordinate ascent procedure which performs two alternative steps:
E-step: Compute the upper bound $u(\boldsymbol{\theta}, \boldsymbol{\theta}^r) \triangleq \mathbb{E}_{\mathbf{z}|\mathbf{w},\boldsymbol{\theta}^r}\{\ln\, p(\mathbf{w},\mathbf{z}|\theta)\}$
M-step: Optimize the upper bound i.e. $\boldsymbol{\theta}^{r+1} = \arg\max_{\boldsymbol{\theta}} u(\boldsymbol{\theta},\boldsymbol{\theta}^r)$

Note: EM algorithm is an example of a class of algorithms called Majorization-Minimization or Convex-Concave Procedure (CCCP) or Difference of Convex Programming (DC Programming). These algorithms aim to minimize the

sum of a convex and a concave function and do so by creating a first-order taylor approximation for the concave function, which gives an overall convex upper bound for the whole objective function.

4.3 Transcript Abundance Estimation

This problem setup involves M sequences which can be read via a reading mechanism. The sequences are intermingled and cannot be directly accessed, except via our blind reading mechanism which draws a sequence s_m according to its occurence probability ρ_m and generates a reading R_n at the n^{th} attempt. Every time the mechanism reads, it incurs small errors and returns back an erroneous version of the actual sequnce read s.t. the probability of observing a particular reading R_n from the true sequence s_m is given by α_{nm}.

Given N readings $R_{n=1:N}$ and assuming that all sequnces $s_{mm=1:M}$ are known, we want to estimate their occurence probability ρ_m. We can write the joint probability of all readings as:

$$\begin{aligned} Pr(R_1, \ldots, R_N; \rho_1, \ldots, \rho_M) &= \prod_{n=1}^{N} Pr(R_n; \rho_1, \ldots, \rho_M) \\ &= \prod_{n=1}^{N} \left(\sum_{m=1}^{M} Pr(R_n | \text{read } R_n \text{ from sequence } s_m) Pr(s_m) \right) \\ &= \prod_{n=1}^{N} \left(\sum_{m=1}^{M} \alpha_{nm} \rho_m \right) \end{aligned}$$

Computing the maximum-likelihood estimate requires solving the problem:

$$\begin{aligned} \hat{\rho_{ML}} = \arg\min_{\rho} \quad & -\sum_{n=1}^{N} \log \left(\sum_{m=1}^{M} \alpha_{nm} \rho_m \right) \\ \text{s.t.} \quad & \sum_{m=1}^{M} \rho_m = 1, \text{ and } \rho_m \geq 0, \forall m \in [M] \end{aligned}$$

Since this cannot be done directly, we create an upper bound per iteration:

$$\rho^{r+1} = \arg\min_{\rho} \quad -\sum_{n=1}^{N}\left(\sum_{m=1}^{M}\left(\frac{\alpha_{nm}\rho_m^r}{\sum_{m'=1}^{M}\alpha_{nm'}\rho_{m'}^r}\log\left(\frac{\rho_m}{\rho_m^r}\right)\right)+\log\left(\sum_{m=1}^{M}\alpha_{nm}\rho_m^r\right)\right)$$
$$\text{s.t.} \sum_{m=1}^{M}\rho_m = 1, \text{ and } \rho_m \geq 0, \forall m \in [M]$$

Then the upper bound can be minimized with a closed-form update rule:

$$\rho_m^{r+1} = \frac{1}{N}\sum_{n=1} N\frac{\alpha_{nm}\rho_m^r}{\sum_{m'=1}^{M}\alpha_{nm'}\rho_{m'}^r}, \forall m \in [M]$$

References

[1] D. P. Bertsekas "Nonlinear Programming", *Belmont: Athena scientific*, 1999.

[2] Tseng, P. "Convergence of a Block Coordinate Descent Method for Nondifferentiable Minimization", *Journal of Optimization Theory and Applications*, 2001.

[3] Håstad, Johan, "Tensor Rank is NP-complete", J. Algorithms, Dec. 1990.