Αλγοριθμικές Μέθοδοι Βελτιστοποίησης με Έμφαση σε Κατανεμημένα Προβλήματα

Βασικές Τεχνικές Κατανεμημένης Βελτιστοποίησης (μέρος 2)

Δημήτρης Αμπελιώτης

Επίκουρος Καθηγητής, Ιόνιο Πανεπιστήμιο

Περιεχόμενα

- Κατανεμημένος Αλγόριθμος Κλίσεων
- Αλγόριθμος Δυικής Ανόδου (Dual Ascent)
- Alternating Direction Method of Multipliers
- Μέθοδοι Δεύτερης Τάξης

Κατανεμημένος Αλγόριθμος Κλίσεων

(Αντιστοιχεί στην Παράγραφο 5.3 στο “Cooperative and Graph Signal Processing: Principles and Applications”)

Κατανεμημένος Αλγόριθμος Κλίσεων

- Όπως είδαμε στα προηγούμενα, η μέθοδος της χαλάρωσης στον πρωτεύοντα χώρο (primal relaxation, γνωστή και ως μέθοδος εισαγωγής συναρτήσεων ποινής – penalty function method) ορίζει το πρόβλημα βελτιστοποίησης

$$\boldsymbol{x}_a^* = \underset{\boldsymbol{x}\in\mathbb{R}^{n\cdot p}}{\operatorname{argmin}} \left(\alpha \cdot F(\boldsymbol{x}) + \frac{1}{2} \|\mathbf{B}\mathbf{x}\|^2 \right)$$

- όπου
  - $F(\boldsymbol{x}) = \sum_{i=1}^{n} f_i(x_i)$
  - $\boldsymbol{x} = [x_1^T \quad x_2^T \quad \cdots \quad x_n^T]^T \in \mathbb{R}^{n\cdot p}$
  - $x_i \in \mathbb{R}^p$
  - $\mathbf{B} = B \otimes \mathbf{I}_\mathrm{p}$
  - $B_{e,v} = B_{(i,j),v} = \begin{cases} \sqrt{w_{ij}}, & \alpha\nu\ v = i \\ -\sqrt{w_{ij}}, & \alpha\nu\ v = j \\ 0, & \delta\iota\alpha\varphi o\rho\varepsilon\tau\iota\kappa\alpha \end{cases}$

$$\mathbf{A} \otimes \mathbf{B} = \begin{bmatrix} a_{11}\mathbf{B} & \cdots & a_{1n}\mathbf{B} \\ \vdots & \ddots & \vdots \\ a_{m1}\mathbf{B} & \cdots & a_{mn}\mathbf{B} \end{bmatrix}$$

Κατανεμημένος Αλγόριθμος Κλίσεων

- Ας δούμε τι μορφή λαμβάνει ο (κεντρικοποιημένος) αλγόριθμος gradient descend για την επαναληπτική επίλυση του κυρτού προβλήματος

$$\boldsymbol{x}_a^* = \underset{\boldsymbol{x}\in\mathbb{R}^{n\cdot p}}{\operatorname{argmin}} \left(\alpha \cdot F(\boldsymbol{x}) + \frac{1}{2}\|\mathbf{B}\mathbf{x}\|^2 \right)$$

- Ορίζουμε:
  - $f_a(\boldsymbol{x}) \coloneqq aF(\boldsymbol{x}) + \frac{1}{2}\|\mathbf{B}\mathbf{x}\|^2$
  - $\boldsymbol{x}_k$ είναι το διάνυσμα το οποίο διατηρεί ο αλγόριθμος την επανάληψη k
  - $\boldsymbol{g}_k \coloneqq \nabla f_a(\boldsymbol{x}_k)$ είναι το διάνυσμα κλίσεων (gradient vector)

- Έτσι, έχουμε πως
  - $\boldsymbol{g}_k = \alpha\nabla F(\boldsymbol{x}_k) + \mathbf{B}^\mathbf{T}\mathbf{B}\boldsymbol{x}_k = \alpha\nabla F(\boldsymbol{x}_k) + \mathbf{L}\,\boldsymbol{x}_k$, όπου $\mathbf{L} = \mathbf{B}^\mathbf{T}\mathbf{B} = L \otimes \mathbf{I}_\mathrm{p}$

- Έτσι, ο αλγόριθμος gradient descend παίρνει τη μορφή:

$$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k - \epsilon_k \boldsymbol{g}_k = \boldsymbol{x}_k - \epsilon_k[a\nabla F(\boldsymbol{x}_k) + \mathbf{L}\boldsymbol{x}_k]$$

Κατανεμημένος Αλγόριθμός Κλίσεων

- Εκφράζουμε το διάνυσμα κλίσεων $\boldsymbol{g}_k \in \mathbb{R}^{n \cdot p}$ ως

$$\boldsymbol{g}_k = \begin{bmatrix} g_{1,k}^T & g_{2,k}^T & \cdots & g_{n,k}^T \end{bmatrix}^T$$

όπου $g_{i,k}^T \in \mathbb{R}^p$ είναι ένα τμήμα του διανύσματος κλίσεων που κρατά μόνο τις κλίσεις ως προς το x_i στην επανάληψη k

- Μπορούμε να δείξουμε πως

$$g_{i,k} = a\nabla f_i\left(x_{i,k}\right) + x_{i,k} - \sum_{j \in n(i)} w_{ij} x_{j,k}$$

- Παρατηρούμε πως το $g_{i,k}$ μπορεί να υπολογιστεί τοπικά στον κόμβο i, αρκεί ο κόμβος να έχει στη διάθεσή του τα $x_{j,k}$, δηλαδή τα τοπικά διανύσματα των μεταβλητών από όλους τους γειτονικούς του κόμβους

Κατανεμημένος Αλγόριθμος Κλίσεων

- Γράφουμε τη συνολική εξίσωση ανανέωσης ως

$$\begin{bmatrix} x_{1,k+1} \\ x_{2,k+1} \\ \vdots \\ x_{n.k+1} \end{bmatrix} = \begin{bmatrix} x_{1,k} \\ x_{2,k} \\ \vdots \\ x_{n.k} \end{bmatrix} - \epsilon_k \begin{bmatrix} a\nabla f_1(x_{1,k}) + x_{1,k} - \sum_{j\in n(1)} w_{1j}x_{j,k} \\ a\nabla f_2(x_{2,k}) + x_{2,k} - \sum_{j\in n(2)} w_{2j}x_{j,k} \\ \vdots \\ a\nabla f_n(x_{n,k}) + x_{n,k} - \sum_{j\in n(n)} w_{nj}x_{j,k} \end{bmatrix}$$

- Αν όλοι οι κόμβοι ανανεώσουν το τοπικό διάνυσμα των μεταβλητών τους, τότε αυτό **θα είναι ισοδύναμο με μια επανάληψη του κεντρικοποιημένου αλγορίθμου gradient descend!** (στον np-διάστατο χώρο, όχι στον αρχικό)
- Έχουμε έτοιμα όλα τα αποτελέσματα σύγκλισης που ισχύουν για τον κεντρικοποιημένο αλγόριθμο
- Οι κόμβοι ανανεώνουν με οποιαδήποτε σειρά, ακόμα και **παράλληλα!**

Κατανεμημένος Αλγόριθμος Κλίσεων

Συνολικά ο Αλγόριθμος (από το βιβλίο)

Algorithm 5.1 DISTRIBUTED GRADIENT DESCENT (DGD) METHOD AT NODE *i*

Require: Initial iterate $x_{i,0}$, weights w_{ij}, penalty coefficient α, step sizes ϵ_k.

1: **for** $k = 0, 1, 2, \ldots$ **do**

2: Exchange iterates $x_{i,k}$ with neighbors $j \in n(i)$.

3: Evaluate gradient $g_{i,k} = \alpha \nabla f_i(x_{i,k}) + x_{i,k} - \sum_{j \in n(i)} w_{ij} x_{j,k}$.

4: Update local iterate: $x_{i,k+1} = x_{i,k} - \epsilon_k \, g_{i,k}$.

5: **end for**

- Αν η συνάρτηση $f_a(\cdot)$ είναι «ισχυρά κυρτή» (strongly convex), τότε ο αλγόριθμος συγκλίνει με σταθερό $\epsilon_k = \epsilon$ (αρκεί να είναι «αρκετά» μικρό)
- Ο ρυθμός σύγκλισης είναι γραμμικός

- Σημείωση: Πρέπει η συνάρτηση κόστους να είναι παραγωγίσιμη!

Κατανεμημένος Αλγόριθμος Κλίσεων

Μια Ενδιαφέρουσα Σημείωση

- Για σταθερό ϵ και ορίζοντας $\tilde{w}_{ii} = (1-\epsilon), \tilde{w}_{ij} = \epsilon\, w_{ij}$ και $\tilde{a} = \epsilon a$, έχουμε

$$\begin{bmatrix} x_{1,k+1} \\ x_{2,k+1} \\ \vdots \\ x_{n.k+1} \end{bmatrix} = \begin{bmatrix} \sum_{j \in n(1) \cup \{1\}} \tilde{w}_{1j} x_{j,k} - \tilde{a} \nabla f_1(x_{1,k}) \\ \sum_{j \in n(2) \cup \{2\}} \tilde{w}_{2j} x_{j,k} - \tilde{a} \nabla f_2(x_{2,k}) \\ \vdots \\ \sum_{j \in n(n) \cup \{n\}} \tilde{w}_{nj} x_{j,k} - \tilde{a} \nabla f_n(x_{n,k}) \end{bmatrix}$$

το οποίο έχει την ενδιαφέρουσα μορφή:

- Υπολογισμός μέσου όρου των γειτονικών μεταβλητών
- Ανανέωση σύμφωνα με την τοπική συνάρτηση κόστους

Αλγόριθμος Δυικής Ανόδου

(Αντιστοιχεί στην Παράγραφο 5.4.1 στο “Cooperative and Graph Signal Processing: Principles and Applications”)

Αλγόριθμός Δυικής Ανόδου

- Όπως είδαμε στα προηγούμενα, η μέθοδος της «χαλάρωσης στο δυικό χώρο» (dual constraint relaxation) ακολουθεί τη μεθοδολογία των πολλαπλασιαστών Lagrange. Πιο συγκεκριμένα
  - Η εξίσωση περιορισμών $\mathbf{Bx} = \mathbf{0}$ «ενσωματώνεται» στη Lagrangian συνάρτηση, εισάγοντας επίσης ένα διάνυσμα πολλαπλασιαστών Lagrange $\boldsymbol{v} \in \mathbb{R}_{+}^{m \cdot p}$

$$\mathcal{L}(\mathbf{x}, \boldsymbol{v}) = F(\mathbf{x}) + \boldsymbol{v}^T \mathbf{Bx}$$

  - Έστω πως οι βέλτιστες τιμές του $\mathbf{x}$ για δοσμένες τιμές του $\boldsymbol{v}$ δίνονται από τη σχέση

$$\mathbf{x}(\boldsymbol{v}) = \underset{\mathbf{x} \in \mathbb{R}^{n \cdot p}}{\operatorname{argmin}} \mathcal{L}(\mathbf{x}, \boldsymbol{v})$$

  - και οι αντίστοιχες τιμές της Lagrangian ορίζουν τη «δυική συνάρτηση»

$$g(\boldsymbol{v}) = \min_{\mathbf{x} \in \mathbb{R}^{n \cdot p}} \mathcal{L}(\mathbf{x}, \boldsymbol{v}) = \mathcal{L}(\mathbf{x}(\boldsymbol{v}), \boldsymbol{v}) = F\big(\mathbf{x}(\boldsymbol{v})\big) + \boldsymbol{v}^T \mathbf{Bx}(\boldsymbol{v})$$

  - Το πρόβλημα λύνεται από τη μεγιστοποίηση της δυικής συνάρτησης. Η βέλτιστες τιμές για τους πολλαπλασιαστές Lagrange είναι το όρισμα που μεγιστοποιεί τη δυική συνάρτηση

$$\boldsymbol{v}^* = \underset{\boldsymbol{v} \in \mathbb{R}^{m \cdot p}}{\operatorname{argmax}} g(\boldsymbol{v})$$

Αλγόριθμός Δυικής Ανόδου

- Επαναληπτική επίλυση
  - Θεωρούμε πως στην επανάληψη k έχουμε τους πολλαπλασιαστές $\boldsymbol{v}^{k-1}$ από την προηγούμενη επανάληψη (αρχικοποιούμε σε κάποιο διάνυσμα $\boldsymbol{v}^0$)
  - Τότε, για $k = 1, 2, \ldots$
    - Βρίσκουμε τις τιμές του διανύσματος $\boldsymbol{x}$ που ελαχιστοποιούν την Lagrangian για τις συγκεκριμένες τιμές των πολλαπλασιαστών
$$\boldsymbol{x}^k = \underset{\boldsymbol{x}\in\mathbb{R}^{np}}{\operatorname{argmin}} \mathcal{L}\left(\mathbf{x}, \boldsymbol{v}^{k-1}\right) = g\left(\boldsymbol{v}^{k-1}\right)$$
    - Διαπιστώνουμε πως το σημείο $\left(\boldsymbol{v}^{k-1}, g\left(\boldsymbol{v}^{k-1}\right)\right) = \left(\boldsymbol{v}^{k-1}, \mathcal{L}\left(\mathbf{x}^k, \boldsymbol{v}^{k-1}\right)\right)$ ανήκει στη δυική συνάρτηση
    - Χρησιμοποιούμε τον αλγόριθμο κλίσης για να ανανεώσουμε τις τιμές των πολλαπλασιαστών, με σκοπό να αυξήσουμε την τιμή της δυικής συνάρτησης
$$\boldsymbol{v}^k = \boldsymbol{v}^{k-1} - \eta_k \mathbf{B}\boldsymbol{x}^k$$
    - όπου $\{\eta_k\}_{k\geq 1}$ μια ακολουθία από θετικά βήματα

Αλγόριθμός Δυικής Ανόδου

- Θα δούμε πως ο αλγόριθμος αυτός μπορεί να υλοποιηθεί επ' ακριβώς με κατανεμημένο τρόπο
- Για να το δείξουμε αυτό, αρχικά ορίζουμε για κάθε κόμβο i δύο σύνολα γειτονικών κόμβων:
  - $L_{i,+}$ που περιέχει τους γειτονικούς κόμβους του i με $\mathrm{i} > j$
  - $L_{i,-}$ που περιέχει τους γειτονικούς κόμβους του i με $\mathrm{i} < \mathrm{j}$
- Χρησιμοποιώντας αυτά τα σύνολα κόμβων, μπορούμε να γράψουμε

$$\mathcal{L}(\mathbf{x}, \boldsymbol{v}) = F(\mathbf{x}) + \boldsymbol{v}^T \mathbf{B}\mathbf{x}$$

$$\mathcal{L}(\mathbf{x}, \boldsymbol{v}) = \sum_{i=1}^{n} f_i(x_i) - \sum_{l=1}^{m} v_l^T \left(x_{i_l} - x_{j_l}\right)$$

$$\mathcal{L}(\mathbf{x}, \boldsymbol{v}) = \sum_{i=1}^{n} \left(f_i(x_i) + x_i^T \left(\sum_{l \in L_{i,+}} v_l - \sum_{l \in L_{i,-}} v_l\right)\right)$$

- Από την τελευταία μορφή, φαίνεται πως όταν οι συντελεστές Lagrange διατηρούνται σταθεροί, τότε η συνάρτηση $\mathcal{L}\left(\mathbf{x}, \boldsymbol{v}^k\right)$ είναι διαχωρίσιμη ως προς τα διανύσματα $x_1, x_2, \ldots, x_n$
- Έτσι, η ελαχιστοποίηση μπορεί να γίνει ανεξάρτητα σε κάθε κόμβο

Αλγόριθμός Δυικής Ανόδου

- Επαναληπτική κατανεμημένη επίλυση
  - Αρχικοποιούμε τους πολλαπλασιαστές Lagrange σε ένα διάνυσμα $\boldsymbol{v}^0$, έτσι ώστε για κάθε ακμή $e_l = (i_l, j_l)$ του γραφήματος, οι κόμβοι i_l και j_l να συμφωνούν στις ίδιες τιμές v_l^0
  - Για $k = 1, 2, \ldots$
    - Βρίσκουμε τις τιμές του διανύσματος $\boldsymbol{x}$ που ελαχιστοποιούν την Lagrangian για τις συγκεκριμένες τιμές των πολλαπλασιαστών. Κάθε κόμβος ανεξάρτητα λύνει το πρόβλημα

$$x_i^k = \underset{x \in \mathbb{R}^p}{\operatorname{argmin}} \left(f_i(x) + x^T \left(\sum_{l \in L_{i,+}} v_l^{k-1} - \sum_{l \in L_{i,-}} v_l^{k-1} \right) \right)$$

    - Οι γειτονικοί κόμβοι ανταλλάσσουν τις εκτιμήσεις τους x_i^k
    - Οι κόμβοι ανανεώνουν τους πολλαπλασιαστές Lagrange σύμφωνα με τον αλγόριθμο κλίσης

$$\boldsymbol{v}^k = \boldsymbol{v}^{k-1} - \eta_k \boldsymbol{B}\boldsymbol{x}^k$$

    - όπου $\{\eta_k\}_{k \geq 1}$ μια ακολουθία από θετικά βήματα

Αλγόριθμός Δυικής Ανόδου

- Παρατηρήσεις
  - Ο αλγόριθμος δυικής ανόδου σέβεται τους περιορισμούς για επικοινωνία μόνο μέσω των ακμών του γραφήματος
  - Στη περίπτωση όπου οι τοπικές συναρτήσεις κόστους είναι ισχυρά κυρτές ο αλγόριθμος είναι σε θέση να μας δώσει την ακριβή λύση (αυτή που θα υπολογίζαμε αν λύναμε κεντρικοποιημένα το αρχικό πρόβλημα), αρκεί να ισχύουν επιπλέον κάποιες συνθήκες
  - Αυτές οι επιπλέον συνθήκες, ωστόσο, είναι κάπως περιοριστικές. Το πρόβλημα αυτό λύνει ο επόμενος αλγόριθμος.

Alternating Direction Method of Multipliers (ADMM)

(Αντιστοιχεί στην Παράγραφο 5.4.2 στο “Cooperative and Graph Signal Processing: Principles and Applications”)

Alternating Direction Method of Multipliers

- Για να ξεπεράσουμε κάποιους περιορισμούς της μεθόδου δυικής ανόδου, η οποία βασίζεται στη Lagrangian συνάρτηση, ορίζουμε εδώ μια παραλλαγή της, την οποία καλούμε «Επαυξημένη Lagrangian Συνάρτηση» (Augmented Lagrangian)
- Η επαυξημένη Lagrangian συνάρτηση εισάγει και τετραγωνικούς όρους σε σχέση με τους περιορισμούς συναίνεσης

$$\mathcal{L}_\rho(\mathbf{x}, \boldsymbol{v}) = \sum_{i=1}^{n} f_i(x_i) + \boldsymbol{v}^T \mathbf{B}\mathbf{x} + \frac{\rho}{2} \sum_{(i,j)\in E} \|x_i - x_j\|_2^2$$

  - όπου $\rho > 0$ μια σταθερά

- Παρατηρούμε πως
  - Για όλα τα διανύσματα $\mathbf{x}$ που ικανοποιούν τους περιορισμούς συναίνεσης, ο επιπλέον τετραγωνικός όρος μηδενίζεται
  - Ωστόσο, η εισαγωγή του τετραγωνικού όρου μπορεί να μετατρέψει την επαυξημένη Lagrangian σε μια ισχυρά κυρτή συνάρτηση, ακόμα και όταν οι επιμέρους συναρτήσεις κόστους δεν είναι ισχυρά κυρτές

Alternating Direction Method of Multipliers

- Ένας (κεντρικοποιημένος) αλγόριθμος για βελτιστοποίηση με χρήση της επαυξημένης Lagrangian συνάρτησης είναι *η μέθοδος των πολλαπλασιαστών (method of multipliers)*
- Σύμφωνα με τη μέθοδο αυτή, έχουμε επαναλήψεις της μορφής

$$\boldsymbol{x}^k = \underset{\boldsymbol{x}\in\mathbb{R}^{np}}{\operatorname{argmin}} \mathcal{L}_p\left(\mathbf{x}, \boldsymbol{v}^{k-1}\right)$$

$$\boldsymbol{v}^k = \boldsymbol{v}^{k-1} + \rho\mathbf{B}\boldsymbol{x}^k$$

- Ωστόσο, η εισαγωγή των τετραγωνικών όρων στην $\mathcal{L}_\rho(\mathbf{x}, \boldsymbol{v})$ την κάνει μη-διαχωρίσιμη (σε αντίθεση με την $\mathcal{L}(\mathbf{x}, \boldsymbol{v})$), επομένως δεν μπορούμε να υλοποιήσουμε την ελαχιστοποίηση με κατανεμημένο τρόπο
- Ιδέα για αποφυγή αυτού του coupling:
  - Η ανανέωση του $\boldsymbol{x}^k$ θα γίνεται τμηματικά, όπου κάθε φορά ανανεώνεται το τμήμα του (π.χ. $\boldsymbol{x}_i^k$) που αντιστοιχεί σε έναν κόμβο (π.χ. τον i)
  - Κατά την ανανέωση του $\boldsymbol{x}_i^k$ οι υπόλοιπες μεταβλητές θα έχουν **την πιο πρόσφατη τιμή τους** (ερώτηση: είναι το ίδιο ;)

Alternating Direction Method of Multipliers

- Για να περιγράψουμε την ιδέα αυτή με μεγαλύτερη ακρίβεια, ορίζουμε
$$\mathcal{L}_{i,\rho}(x_i, \tilde{\boldsymbol{x}}_{-i}, \boldsymbol{v}) = \mathcal{L}_p([\tilde{x}_1 \quad \tilde{x}_2 \quad \cdots \quad \tilde{x}_{i-1} \; x_i \; \tilde{x}_{i+1} \cdots \quad \tilde{x}_n], \boldsymbol{v})$$
  - όπου το όρισμα x_i καθορίζει τις μεταβλητές του κόμβου i, ενώ το όρισμα $\tilde{\boldsymbol{x}}_{-i}$ καθορίζει τις μεταβλητές όλων των υπολοίπων κόμβων

Επαναληπτική κατανεμημένη επίλυση

- Αρχικοποιούμε τους πολλαπλασιαστές Lagrange σε ένα διάνυσμα $\boldsymbol{v}^0$, έτσι ώστε για κάθε ακμή $e_l = (i_l, j_l)$ του γραφήματος, οι κόμβοι i_l και j_l να συμφωνούν στις ίδιες τιμές v_l^0. Επίσης, κάθε κόμβος αρχικοποιεί τις μεταβλητές του x_i^0 και ενημερώνει τους γείτονές του
- Για $k = 1, 2, \ldots$
  - Για $i = 1, 2, \ldots n$
    - $x_i^k = \underset{\boldsymbol{x} \in \mathbb{R}^p}{\operatorname{argmin}} \mathcal{L}_{i,p}(x, \tilde{\boldsymbol{x}}_{-i}^{k-1}, \boldsymbol{v}^{k-1})$
    - Στείλε το x_i^k στους γείτονες οι οποίοι ενημερώνουν το $\tilde{\boldsymbol{x}}_{-i}^{k-1}$
  - Οι κόμβοι ανανεώνουν τους πολλαπλασιαστές Lagrange σύμφωνα με τον αλγόριθμο κλίσης
    - $\boldsymbol{v}^k = \boldsymbol{v}^{k-1} + \rho \boldsymbol{B}\boldsymbol{x}^k$

Alternating Direction Method of Multipliers

- Παρατηρήσεις
  - Η σειρά των κόμβων με την οποία θα λυθούν τα υπό-προβλήματα στην εσωτερική επανάληψη $x_i^k = \underset{\boldsymbol{x}\in\mathbb{R}^p}{\operatorname{argmin}} \mathcal{L}_{i,p}\left(x, \tilde{\boldsymbol{x}}_{-i}^{k-1}, \boldsymbol{v}^{k-1}\right)$ είναι αυθαίρετη
  - Η σειρά αυτή μπορεί να αλλάζει από εξωτερική επανάληψη σε εξωτερική επανάληψη
  - Επίσης, η σειρά μπορεί να καθορίζεται τυχαία, αρκεί ασυμπτοτικά κάθε κόμβος να ανανεώνει τις μεταβλητές του το ίδιο συχνά
    - π.χ. διαλέγω τον επόμενο κόμβο που θα ανανεώσει με τυχαίο τρόπο από όλο το δίκτυο, με ίσες πιθανότητες. Μπορεί αυτό να γίνει στην πράξη;
  - Όλες αυτές οι παραλλαγές έχουν μελετηθεί και έχει βρεθεί πως έχουν γραμμική σύγκλιση, όταν οι τοπικές συναρτήσεις κόστους είναι κυρτές (όχι απαραίτητα ισχυρά κυρτές)

  - **Σημείωση:** Είδαμε πως αυτή η τμηματική ανανέωση των μεταβλητών x_i^k δεν είναι ισοδύναμη με την ανανέωση ολόκληρου του διανύσματος $\boldsymbol{x}^k$ την οποία «απαιτούσε» η κεντρικοποιημένη μέθοδος. Ωστόσο, είπαμε πως η παραλλαγή αυτή συγκλίνει. Πως είναι δυνατόν αυτό;
  - Σε διάφορες εργασίες προτείνουν διάφορες παραλλαγές (π.χ. Surrogates)

Alternating Direction Method of Multipliers

Μια τελευταία σημείωση

- Η συνάρτηση η οποία ελαχιστοποιείται σε κάθε κόμβο γράφεται

$$\mathcal{L}_{i,p}\left(x_i, \tilde{\boldsymbol{x}}_{-i}^{k-1}, \boldsymbol{v}^{k-1}\right) = f_i(x_i) + x_i^T\left(\sum_{l\in L_{i,+}} v_l^{k-1} - \sum_{l\in L_{i,-}} v_l^{k-1}\right)$$
$$+\frac{\rho}{2}\sum_{j\in n(i)}\left\|x_i - \tilde{x}_j\right\|_2^2 + c_i\left(\tilde{\boldsymbol{x}}_{-i}^{k-1}, \boldsymbol{v}^{k-1}\right)$$

  - όπου ο τελευταίος όρος συνοψίζει όρους που εξαρτώνται από κόμβους που δεν είναι γείτονες του κόμβου i και από πολλαπλασιαστές οι οποίοι αντιστοιχούν σε ακμές όχι προσκείμενες στον κόμβο i, επομένως είναι σταθερός ως προς το x_i και μπορεί να παραληφθεί
  - Έτσι, τα προβλήματα αυτά μπορούν να λύνονται ανεξάρτητα σε κάθε κόμβο

Μέθοδοι Δεύτερης Τάξης

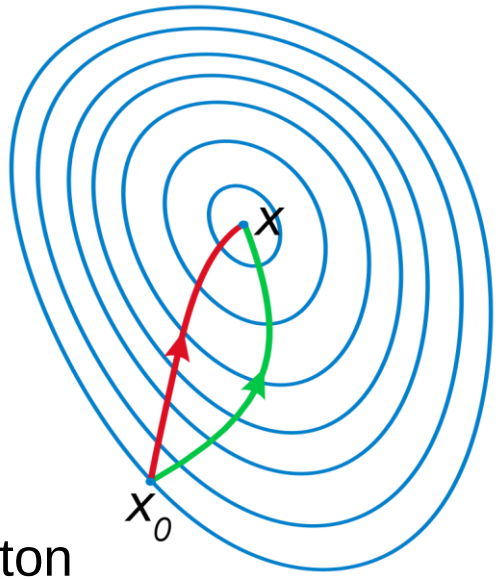
(Αντιστοιχεί στην Παράγραφο 5.5 στο “Cooperative and Graph Signal Processing: Principles and Applications”)

Μέθοδοι Δεύτερης Τάξης

Υπενθύμιση: Μέθοδος Newton

- Ελαχιστοποιεί επαναληπτικά μια συνάρτηση
- Σε κάθε σημείο x_k θεωρεί το ανάπτυγμα Taylor

$$f(x_k + t) \approx f(x_k) + f'(x_k) \cdot t + \frac{1}{2} f''(x_k) \cdot t^2$$



- Ορίζει ως επόμενο σημείο x_{k+1} το σημείο που ελαχιστοποιεί αυτή τη δευτεροβάθμια εξίσωση

$$x_{k+1} = x_k - \frac{f'(x_k)}{f''(x_k)}$$

- Υπάρχουν επίσης οι εκδοχές damped/relaxed Newton στις οποίες έχουμε

$$x_{k+1} = x_k - \gamma \frac{f'(x_k)}{f''(x_k)}, \qquad \gamma < 1$$

- Σε μεγαλύτερες διαστάσεις έχουμε τη γενίκευση

$$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k - [\nabla^2 f(\boldsymbol{x}_k)]^{-1} \nabla f(\boldsymbol{x}_k)$$

Μέθοδοι Δεύτερης Τάξης

- Μπορούμε να χρησιμοποιήσουμε τη μέθοδο Newton για να ελαχιστοποιήσουμε την συνάρτηση κόστους που χρησιμοποιεί «χαλάρωση στον πρωτεύοντα χώρο»

$$f_a(\boldsymbol{x}) = aF(\boldsymbol{x}) + \frac{1}{2}\|\mathbf{B}\mathbf{x}\|^2$$

- Όπως είχαμε δει, το διάνυσμα κλίσεων σε ένα σημείο $\boldsymbol{x}_k$ είναι

$$\boldsymbol{g}_k = \alpha\nabla F(\boldsymbol{x}_k) + \mathbf{B}^{\mathbf{T}}\mathbf{B}\boldsymbol{x}_k = \alpha\nabla F(\boldsymbol{x}_k) + \mathbf{L}\,\boldsymbol{x}_k$$

- Ο λεγόμενος Hessian πίνακας με τις μερικές παραγώγους δεύτερης τάξης θα είναι

$$\mathbf{H}_k = \nabla^2 f_a(\boldsymbol{x}) = a\mathbf{G}_k + \mathbf{L}$$

  - όπου $\mathbf{G}_k \in \mathbb{R}^{np\times np}$ είναι ένας block διαγώνιος πίνακας με $\mathbf{G}_{ii,k} = \nabla^2 f_i\left(x_{i,k}\right) \in \mathbb{R}^{p\times p}$

- Η εξίσωση ανανέωσης του διανύσματος $\boldsymbol{x}_k$ θα είναι

$$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k - \mathbf{H}_k^{-1}\boldsymbol{g}_{\mathrm{k}}$$

- Αν και ο πίνακας $\mathbf{H}_k$ είναι block sparse, ο αντίστροφός του δεν είναι sparse

Μέθοδοι Δεύτερης Τάξης

- Θεωρώ την ακόλουθη διάσπαση του πίνακα $\mathbf{H}_k$

$$\mathbf{H}_k = \mathbf{D}_k - \mathbf{J}_k$$

- όπου $\mathbf{D}_k = \alpha\mathbf{G}_k + \mathbf{I}$, και έτσι χρησιμοποιώντας και την προηγούμενη σχέση $\mathbf{H}_k = a\mathbf{G}_k + \mathbf{L}$ βρίσκουμε πως $\mathbf{J}_k = \mathbf{I} - \mathbf{L}$
- Παρατηρούμε πως όταν οι αρχικές συναρτήσεις κόστους είναι ισχυρά κυρτές, τότε ο πίνακας $\mathbf{G}_k$ θα είναι θετικά ορισμένος (μοναδικό ελάχιστο)
- Παρατηρούμε τότε πως και ο πίνακας $\mathbf{D}_k$ θα είναι θετικά ορισμένος
- Κόλπο

$$\mathbf{H}_k = \mathbf{D}_k - \mathbf{J}_k$$

$$\mathbf{H}_k = \mathbf{D}_k^{1/2}\mathbf{D}_k^{1/2} - \mathbf{D}_k^{1/2}\mathbf{D}_k^{-1/2}\mathbf{J}_k\mathbf{D}_k^{-1/2}\,\mathbf{D}_k^{1/2}$$

$$\mathbf{H}_k = \mathbf{D}_k^{1/2}\left(\mathbf{I} - \mathbf{D}_k^{-1/2}\mathbf{J}_k\mathbf{D}_k^{-1/2}\right)\mathbf{D}_k^{1/2}$$

$$\mathbf{H}_k^{-1} = \mathbf{D}_k^{-1/2}\left(\mathbf{I} - \mathbf{D}_k^{-1/2}\mathbf{J}_k\mathbf{D}_k^{-1/2}\right)^{-1}\mathbf{D}_k^{-1/2}$$

Μέθοδοι Δεύτερης Τάξης

- Έτσι, ο αντίστροφος είναι

$$\mathbf{H}_k^{-1} = \mathbf{D}_k^{-1/2}\left(\mathbf{I} - \mathbf{D}_k^{-1/2}\mathbf{J}_k\mathbf{D}_k^{-1/2}\right)^{-1}\mathbf{D}_k^{-1/2}$$

- Θα χρησιμοποιήσουμε τώρα την ιδιότητα

$$\sum_{l=0}^{\infty}\lambda^l = \frac{1}{1-\lambda}$$

$$\sum_{l=0}^{\infty}\mathbf{X}^l = (\mathbf{I}-\mathbf{X})^{-1}$$

- και έτσι, καταλήγουμε στην έκφραση

$$\mathbf{H}_k^{-1} = \mathbf{D}_k^{-1/2}\sum_{\boldsymbol{l=0}}^{\boldsymbol{\infty}}\left(\mathbf{D}_k^{-1/2}\mathbf{J}_k\mathbf{D}_k^{-1/2}\right)^{\boldsymbol{l}}\mathbf{D}_k^{-1/2}$$

- Τελικά, χρησιμοποιούμε truncated εκδοχές του άπειρου αθροίσματος και ορίζουμε έτσι τις τεχνικές Network Newton - L

Μέθοδοι Δεύτερης Τάξης

Algorithm 5.4 NETWORK NEWTON-*L* METHOD AT NODE *I*

Require: Initial iterate $\mathbf{x}_{i,0}$. Weights w_{ij}. Penalty coefficient α.

J matrix blocks: $\mathbf{J}_{ii} = (1 - w_{ii})\mathbf{I}$ and $\mathbf{J}_{ij} = w_{ij}\mathbf{I}$

for $k = 0, 1, 2, \ldots$ **do**

  D matrix block: $\mathbf{D}_{ii,k} = \alpha\nabla^2 f_i(x_{i,k}) + 2(1 - w_{ii})\mathbf{I}$

  Exchange iterates $x_{i,k}$ with neighbors $j \in n(i)$.

  Gradient: $\mathbf{g}_{i,k} = (1 - w_{ii})x_{i,k} - \sum_{j\in n(i)} w_{ij}\mathbf{x}_{j,k} + \alpha\nabla f_i(x_{i,k})$

  Compute NN-0 descent direction $\mathbf{d}_{i,k}^{(0)} = -\mathbf{D}_{ii,k}^{-1}\mathbf{g}_{i,k}$

  for $l = 0, \ldots, L - 1$ **do**

    Exchange elements $\mathbf{d}_{i,k}^{(l)}$ of the NN-l step with neighbors

    NN-$(l + 1)$ step: $\mathbf{d}_{i,k}^{(l+1)} = \mathbf{D}_{ii,k}^{-1}\left[\sum_{j\in n(i), j=i} \mathbf{J}_{ij}\mathbf{d}_{j,k}^{(l)} - \mathbf{g}_{i,k}\right]$

  end for

  Update local iterate: $\mathbf{x}_{i,k+1} = x_{i,k} + \epsilon\, \mathbf{d}_{i,k}^{(L)}$.

end for

Ερωτήσεις