LAD, Γραμμικός Προγραμματισμός και Αλγόριθμική Επίλυση

Οικονομετρία ΙΙ

Σκοπός:

- Σύντομη περιγραφή του γραμμικού προγραμματισμού.
- ”Πολύ” ευρετική παρουσίαση των αλγορίθμων:
  - **simplex**,
  - **interior-point**.
- Εξήγηση της πιθανής **μη-σύμπτωσης** μεταξύ τους όταν υπάρχουν πολλαπλές βέλτιστες λύσεις.
- Μια απλή άσκηση Monte Carlo για τον LADE.

Από τον LAD στον γραμμικό προγραμματισμό I

Στο γραμμικό υπόδειγμα

$$Y_{(i)} = X'_{(i)}\beta + \varepsilon_{(i)},$$

ο εκτιμητής LAD ορίζεται από το πρόβλημα βελτιστοποίησης

$$\hat{\beta}_{LAD} \in \arg\min_{\beta} \sum_{i=1}^{n} |Y_{(i)} - X'_{(i)}\beta|.$$

Με τη διάσπαση

$$Y_{(i)} - X'_{(i)}\beta = u^{+}_{(i)} - u^{-}_{(i)} := (Y_{(i)} - X'_{(i)}\beta)1_{(Y_{(i)} \geq X'_{(i)}\beta)} - (X'_{(i)}\beta - Y_{(i)})1_{(Y_{(i)} < X'_{(i)}\beta)}, \qquad u^{+}_{(i)}, u^{-}_{(i)} \geq 0,$$

παίρνουμε

$$|Y_{(i)} - X'_{(i)}\beta| = u^{+}_{(i)} + u^{-}_{(i)}.$$

Από τον LAD στον γραμμικό προγραμματισμό II

Άρα το πρόβλημα βελτιστοποίησης, όταν $\Theta = \mathbb{R}^p$, ισοδύναμα αναπαρίσταται από το επαυξημένο πρόβλημα (ως προς $2n + 2p$ μεταβλητές)
(Εδώ έχουμε ότι $\beta^+_{(i)} := \beta_{(i)} 1_{\beta_{(i)} \geq 0}$, $\beta^-_{(i)} := -\beta_{(i)} 1_{\beta_{(i)} < 0}$, συνεπώς $X'_{(i)}\beta = X'_{(i)}\beta^+ - X'_{(i)}\beta^-$):

$$\min_{\beta^+, \beta^-, u^+, u^-} \sum_{i=1}^{n} (u^+_{(i)} + u^-_{(i)})$$

υπό

$$Y_{(i)} - X'_{(i)}\beta^+ + X'_{(i)}\beta^- = u^+_{(i)} - u^-_{(i)}, \qquad u^+_{(i)}, u^-_{(i)} \geq 0.$$

Γενικότερο Συμπέρασμα: εφόσον το Θ περιγράφεται από γραμμικούς περιορισμούς ο LAD αναπαραστάσιμος ως πρόβλημα γραμμικού προγραμματισμού-LP (Linear Programming) .

Τι είναι γραμμικός προγραμματισμός; Ι

Ένα γραμμικό πρόγραμμα έχει τη (κανονική) μορφή (canonical form)

$$\min_{x \in \mathbb{R}^m} c'x$$

υπό γραμμικούς περιορισμούς

$$Ax = b, \qquad x \geq \mathbf{0}$$

(οι ανισότητες μεταξύ διανυσμάτων ερμηνεύονται κατά σημείο).
Στην περίπτωση μας και όταν $\Theta = \mathbb{R}^p$, τότε $m = 2p + 2n$,

$$x = (\beta^{+\prime}, \beta^{-\prime}, u^{+\prime}, u^{-\prime})',$$

$$c = (\mathbf{0}_p', \mathbf{0}_p', \mathbf{1}_n', \mathbf{1}_n')',$$

$$A := \begin{pmatrix} X_n & -X_n & I_n & -I_n \end{pmatrix} \in \mathbb{R}^{n \times (2p+2n)},$$

$$b := Y_n.$$

Βασικά στοιχεία:

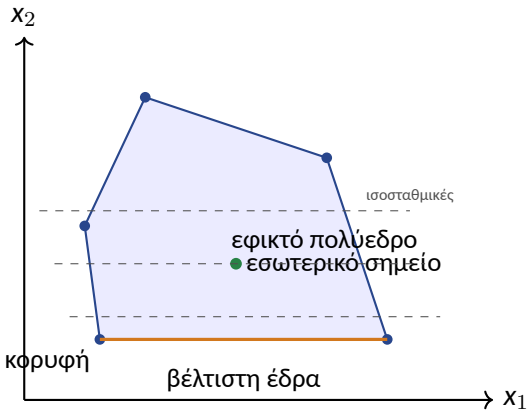
Τι είναι γραμμικός προγραμματισμός; II

- γραμμική αντικειμενική συνάρτηση,
- γραμμικοί περιορισμοί,
- εφικτό σύνολο = πολύεδρο (π.χ. ημιεπίπεδα, κυρτά πολύτοπα, κ.ο.κ.).

Γεωμετρική ιδέα:

- αν υπάρχει μοναδική λύση, συχνά βρίσκεται σε κορυφή,
- αν υπάρχουν πολλαπλές λύσεις, το βέλτιστο καταλαμβάνει συχνά μια ολόκληρη έδρα.

Γεωμετρική εικόνα: το εφικτό σύνολο ως πολύεδρο



- Το σύνολο εφικτών λύσεων ενός LP είναι ένα πολύεδρο.
- Οι κορυφές είναι ειδικά σημεία του πολυέδρου.
- Αν υπάρχουν πολλές βέλτιστες λύσεις, αυτές μπορεί να συγκροτούν ολόκληρη έδρα.

Αλγόριθμοι τύπου Simplex: ευρετική εικόνα

Ιδέα:

- όταν μοναδικό το βελτιστοποιούν σημείο βρίσκεται σε κορυφή,
- άρα κινούμαστε από κορυφή σε κορυφή,
- κάθε pivot (στοιχειώδης μετασχηματισμός γραμμών της μήτρας που αναπαριστά το πρόβλημα) βελτιώνει την αντικειμενική συνάρτηση κινούμενο σε κορυφή που βρίσκεται εγγύτερα στην λύση.

Αποδεικνύεται ότι:

- αν υπάρχει μοναδική λύση, η simplex την εντοπίζει,
- αν υπάρχει βέλτιστη έδρα, η simplex συνήθως επιστρέφει ακραίο σημείο-κορυφή αυτής.
- είναι δυνατόν να είναι ”εκθετικά δαπανηρό”.

Αλγόριθμοι τύπου Interior-point: ευρετική εικόνα

Ιδέα:

- κίνηση στο εσωτερικό του εφικτού συνόλου,
- π.χ. ακολουθώντας κεντρικό μονοπάτι,
- προσεγγίζει λύση χωρίς να κινείται απαραίτητα σε κορυφές.

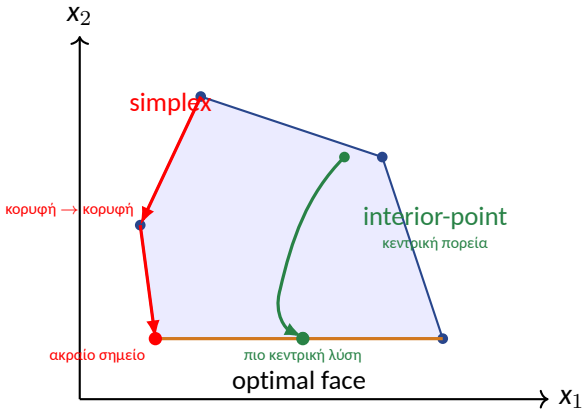
Τυπικά, εφαρμογή βημάτων τύπου Newton-Raphson στο:

$$\min\ c'x - \mu \sum_j \log x_j,\ \text{υπόA}x = b, \qquad \mu \downarrow 0.$$

Επομένως:

- όταν υπάρχουν πολλές λύσεις, η interior-point μέθοδος τείνει να προσεγγίζει το πιο κεντρικό σημείο της βέλτιστης έδρας.

Simplex vs Interior-Point: σχηματική πορεία των αλγορίθμων



- Η simplex κινείται τυπικά πάνω στο σύνορο, από κορυφή σε κορυφή.
- Η interior-point κινείται στο εσωτερικό του πολυέδρου.
- Υπό πλειοτιμία, οι δύο μέθοδοι μπορούν να επιλέξουν διαφορετικά σημεία της βέλτιστης έδρας.

Γιατί μπορεί να διαφέρουν οι αλγόριθμοι;

Οι παραπάνω είναι εφαρμόσιμοι ότον το Θ είναι πολύεδρο (π.χ. το $\mathbb{R}^p$ ή "προσεγγίζεται" από τέτοιο.

Όταν το arg min είναι μονοσύνολο, οι παραπάνω αλγόριθμοι ουσιαστικά συμπίπτουν.

Αν όμως το arg min είναι πλειότιμο:

- simplex: επιλέγει συνήθως ακραίο σημείο,
- interior-point: επιλέγει συνήθως "κεντρικό" εσωτερικό σημείο.

Κεντρική παρατήρηση

Η αριθμητική μέθοδος μπορεί να λειτουργεί ως **κανόνας επιλογής** μέσα στο σύνολο των βέλτιστων λύσεων.

Monte Carlo: βασική κατασκευή

Θεωρούμε το υπόδειγμα

$$Y_{(i)} = (1, z_{(i)})(\beta_0, \beta_1)' + \varepsilon_{(i)}, \qquad \beta_0 = 1,\ \beta_1 = 2.$$

Υπολογίζουμε τον LAD estimator με:

- `highs-ds` (dual simplex),
- `highs-ipm` (interior point).

Ως προς τα δειγματικά μονοπάτια MC ελέγχουμε τα:

$$\|\hat{\beta}_{DS} - \hat{\beta}_{IPM}\|_2$$

και

$$\|\hat{\beta} - \beta_0\|_2.$$

Παρέχουν πληροφορία για την **αλγοριθμική μη-σύμπτωση**, και το **σφάλμα εκτίμησης** αντίστοιχα.

Μονότιμος Σχεδιασμός

$$X_{(i)} = (1, z_{(i)}), \qquad z_{(i)} \sim N(0,1), \qquad \varepsilon_{(i)} \sim \text{Laplace}(0,\sigma); \qquad \text{iid over } i.$$

Χαρακτηριστικά:

- το πληθυσμιακό κριτήριο LAD τυπικά με μοναδικό ελαχιστοποιούν σημείο.

Αναμενόμενο αποτέλεσμα:

- μηδενική ή αμελητέα αλγοριθμική μη-σύμπτωση,
- σταδιακή σύγκλιση προς την αληθή παράμετρο.

Πλειότιμος σχεδιασμός I

$$X_{(i)} = (1, z_{(i)}), \qquad z_{(i)} \in \{-1, 0, 1\}, \qquad \text{iid over } i$$

$$\#\{i : z_{(i)} = -1\} \approx \#\{i : z_{(i)} = 0\} \approx \#\{i : z_{(i)} = 1\} \approx \frac{n}{3}$$

$$\varepsilon_{(i)} \in \{-\sigma, +\sigma\}, \qquad \Pr(\varepsilon_{(i)} = -\sigma) = \Pr(\varepsilon_{(i)} = \sigma) = \tfrac{1}{2}$$

Χαρακτηριστικά:

- διακριτός σχεδιασμός,
- η κατανομή του σφάλματος είναι διακριτή ομοιόμορφη στο $\{-\sigma, \sigma\}$,
- η διάμεσος του $\varepsilon_{(i)}$ δεν είναι μοναδική,
- το sample LAD criterion εμφανίζει ευκολότερα επίπεδες περιοχές.

Πλειότιμος σχεδιασμός II

Συνέπεια για την ταυτοποίηση:

- η απουσία μοναδικής διαμέσου των καταλοίπων καταστρέφει την πληθυσμιακή ταυτοποίηση,
- Πλειότιμα arg min.

Περιμένουμε:

- μη-σύμπτωση των αλγορίθμων σε μικρά δείγματα,
- εξάρτηση της λύσης από τον αλγόριθμο επιλογής,

Code snippet: solver

```
def solve_lad_lp(X, y, method="highs-ds"):
    n, p = X.shape
    m = 2*p + 2*n

    c = np.zeros(m)
    c[2*p:2*p+n] = 1.0
    c[2*p+n:2*p+2*n] = 1.0

    A_eq = np.zeros((n, m))
    b_eq = y.copy()

    A_eq[:, :p] = X
    A_eq[:, p:2*p] = -X
    A_eq[:, 2*p:2*p+n] = np.eye(n)
    A_eq[:, 2*p+n:2*p+2*n] = -np.eye(n)

    bounds = [(0, None)] * m

    res = linprog(c, A_eq=A_eq, b_eq=b_eq,
                  bounds=bounds, method=method)

    z = res.x
    beta = z[:p] - z[p:2*p]
    return beta
```

Code snippet: designs

```python
def simulate_data(n, beta_true, sigma=1.0, seed=None, design="generic"):
    rng = np.random.default_rng(seed)

    if design == "generic":
        X = np.column_stack([np.ones(n), rng.normal(size=n)])
        eps = rng.laplace(0.0, sigma, size=n)
        y = X @ beta_true + eps

    elif design == "degenerate":
        x = np.repeat([-1.0, 0.0, 1.0], repeats=n//3)
        if len(x) < n:
            x = np.concatenate([x, np.zeros(n-len(x))])
        X = np.column_stack([np.ones(n), x])
        eps = rng.choice([-1.0, 1.0], size=n) * sigma
        y = X @ beta_true + eps

    return X, y
```

Περαιτέρω λεπτομέρειες I

Σε κάθε σχεδιασμό χρησιμοποιήθηκε:

- πραγματική παράμετρο
$$\beta_0 = (1, 2)',$$
- μεγέθη δείγματος
$$n \in \{30, 100, 300, 1000\},$$
- αριθμό επαναλήψεων
$$R = 200,$$
- κλίμακα θορύβου
$$\sigma = 1,$$
- επίλυση του ίδιου προβλήματος LAD με δύο λύτες:
  - `highs-ds` (dual simplex),
  - `highs-ipm` (interior point).

Υπολογιστικό κόστος

Για αυτά τα πολύ μικρά LP προβλήματα, ο υπολογιστικός φόρτος είναι εξαιρετικά μικρός.

Σε έναν τυπικό σύγχρονο φορητό υπολογιστή:

- ένα πλήρες Monte Carlo για *έναν* σχεδιασμό (4 τιμές του n, 200 επαναλήψεις, 2 λύτες) τρέχει συνήθως σε μερικά δευτερόλεπτα,
- και οι *δύο* σχεδιασμοί μαζί τυπικά ολοκληρώνονται περίπου σε

$$5\text{–}20 \text{ δευτερόλεπτα},$$

ανάλογα με τον επεξεργαστή και το περιβάλλον Python.

Μονότιμος Σχεδιασμός: κάποια αποτελέσματα:

Απόλυτη σύμπτωση των δύο λύτων:

$$\text{μέση αλγοριθμική απόκλιση} = 0, \qquad \text{διάμεση} = 0, \qquad \text{μέγιστη} = 0$$

για όλα τα n.

Άρα:

- dual simplex και interior point επιστρέφουν ακριβώς τον ίδιο LAD εκτιμητή,
- το sample criterion είναι πρακτικά μονοτιμο σε όλα τα εξεταζόμενα δείγματα.

Ταυτόχρονα το μέσο σφάλμα εκτίμησης φθίνει:

$$0.277253 \;\to\; 0.141340 \;\to\; 0.082401 \;\to\; 0.038418.$$

Ερμηνεία

Στον μονοτιμο σχεδιασμό βλέπουμε καθαρή εμπειρική ένδειξη όχι μόνο ασθενούς συνέπειας, αλλά και ισχυρότερης μορφής σύγκλισης τύπου

$$E[\|\hat{\beta} - \beta_0\|_2] \to 0,$$

Πλειότιμος σχεδιασμός: κάποια αποτελέσματα

n-αλγοριθμική απόκλιση:

$$0.179706,\ 0.116066,\ 0.042426,\ 0.014142$$

για τη μέση αλγοριθμική απόκλιση.

Άρα:

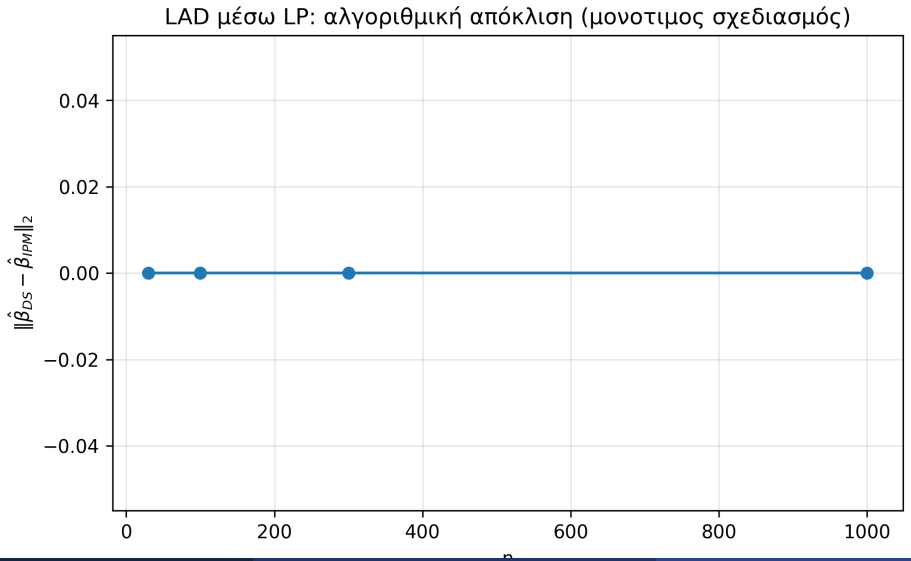
- οι δύο λύτες *δεν* συμπίπτουν σε μικρά δείγματα,
- αλλά η απόκλιση μειώνεται αισθητά όσο το n αυξάνει.

Όμως το μέσο σφάλμα εκτίμησης παραμένει μεγάλο σε όλα τα εξεταζόμενα n.

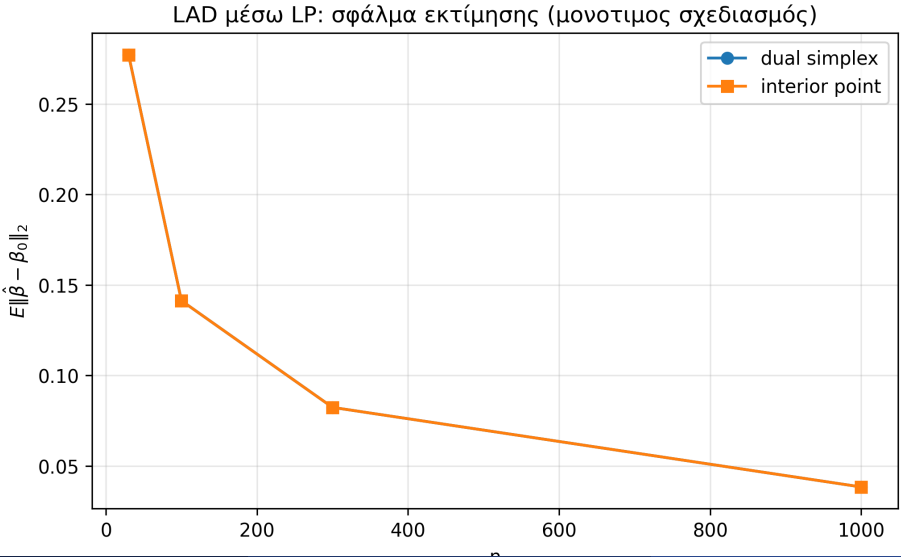
Ερμηνεία

Στον πλειότιμο σχεδιασμό το πείραμα δεν δίνει την ίδια εμπειρική εικόνα συνέπειας. Παρά τη μείωση της αλγοριθμικής απόκλισης, το πειραματικό μέσο σφάλμα εκτίμησης δεν φθίνει.

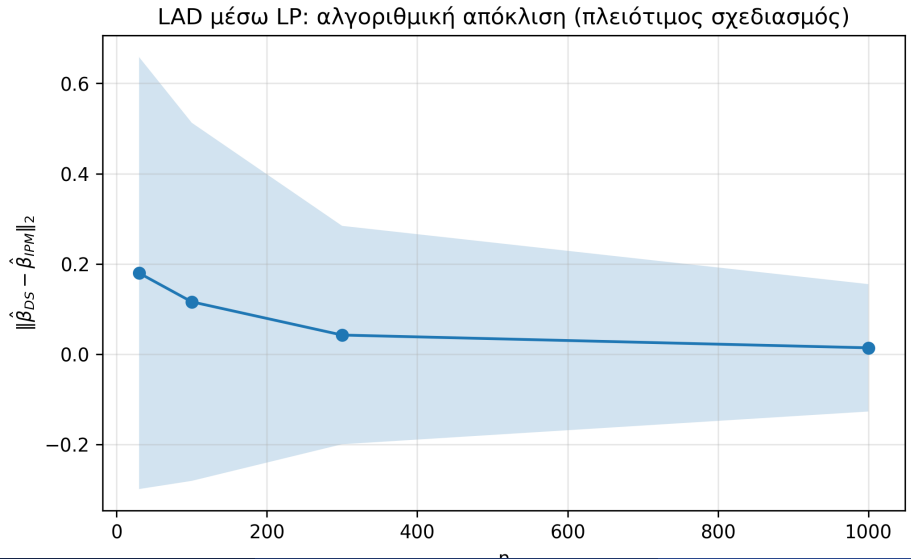
Σχήμα: αλγοριθμική απόκλιση (μονοτιμος σχεδιασμός)



Σχήμα: σφάλμα εκτίμησης (μονοτιμος σχεδιασμός)



Σχήμα: αλγοριθμική απόκλιση (πλειότιμος σχεδιασμός)



Τι μαθαίνουμε από τη σύγκριση των δύο σχεδιασμών;

- **Μονοτιμος σχεδιασμός:**
  - πλήρης σύμπτωση dual simplex και interior point,
  - σαφής πτώση του σφάλματος εκτίμησης.
- **Πλειότιμος σχεδιασμός:**
  - αλγοριθμική απόκλιση σε μικρά δείγματα,
  - η απόκλιση αυτή μειώνεται με το n,
  - αλλά το σφάλμα εκτίμησης δεν ακολουθεί αντίστοιχη καθοδική πορεία.

Επομένως:

- Ο LADE είναι ένα φυσικό παράδειγμα όπου η εκτίμηση απαιτεί αριθμητική βελτιστοποίηση.
- Στο πλειότιμα περιβάλλοντα, οι αλγόριθμοι μπορεί να διαφέρουν συστηματικά ως προς τις λύσεις που επιλέγουν.
- Άρα:

  $$\text{αντικειμενική συνάρτηση} + \text{η γεωμετρία του } \Theta + \text{αλγόριθμος επίλυσης}$$

  συνδιαμορφώνουν τον τελικό υπολογιζόμενο εκτιμητή.

Γενικότερα, η αριθμητική βελτιστοποίηση δεν είναι εργαλείο υπολογισμού. Είναι μέρος του ίδιου του ορισμού του υπολογιζόμενου εκτιμητή, ιδιαίτερα σε περιβάλλοντα μη-μοναδικότητας.