Μάθημα 2: Ενας πράκτορας, Μαρκοβιανές Διαδικασίες Απόφασης, Δυναμικός Προγραμματισμός

Ν. Καλουπτσίδης

Οκτώβριος 2021

Επισκόπηση

- Μαρκοβιανές Διαδικασίες απόφασης, ενας πράκτορας
- Πεπερασμένος χρονικός ορίζοντας και δυναμικός προγραμματισμός (dynamic programming, DP)
- Παράδειγμα: διαχείριση εμπορευμάτων αποθήκης, υλοποίηση στη βιβλιοθήκη quantecon
- Εισαγωγικά στοιχεία για τις νδιαστατες συστοιχίες και το Numpy.
- Απειρος χρονικός ορίζοντας, εξισωση Bellman και υπολογισμός αξίας, αλγόριθμοι σταθερού σημείου.
- Βασικές αλγοριθμικές μέθοδοι:

1. Επανάληψη αξίας (value iteration)
2. Επανάληψη πολιτικής (policy iteration)
3. Τροποποιημένη επανάληψη πολιτικής, ασύγχρονες εκδοχές
4. Γραμμικός προγραμματισμός

Μαρκοβιανές διαδικασίες απόφασης

- μια μόνο μονάδα απόφασης (decision making unit/agent).
- Η μονάδα παρατηρεί πλήρως το περιβάλλον: y=x
- t: χρονική στιγμή, t=1,2, … T,
- T: προκαθορισμένος χρονικός ορίζοντας (επεισόδιο)
- x_t κατάσταση τη στιγμή t; ανήκει στο καταστατικός χώρο $\mathcal{X}$
- a_t ενέργεια, ανήκει στο $\mathcal{A}$ το οποίο μπορεί να εξαρτάται απο τη κατάσταση, $\mathcal{A}$(x), δηλώνει περιορισμούς στις ενέργειες ανάλογα με την κατάσταση.

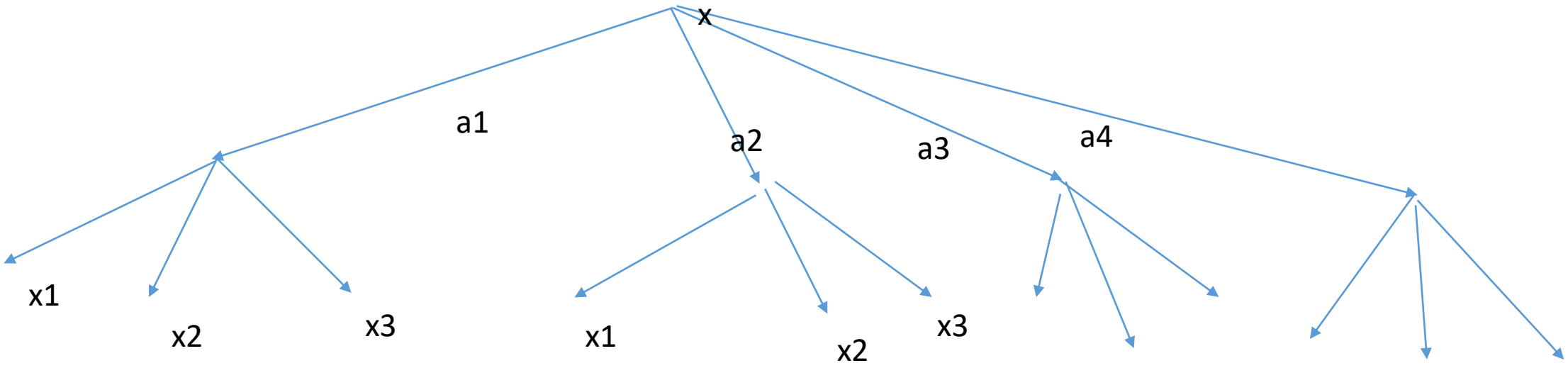
Μαρκοβιανές διαδικασίες απόφασης

- Καταστατική εξέλιξη: δύο ισοδύναμα μοντέλα:

1. $x_{t+1}= f_{t+1}(x_t, a_t, w_t)$, w_t i.i.d θόρυβος, ή
2. $P(x'|x,a)$:=P[$x_{t+1} = \mathrm{x}'|x_t$=x, a_t=a]

- Πράγματι
- $1 \rightarrow 2$:
- $P(x'|x,a) = \sum_w P_w(w) 1_{[x'=f(x,a,w)]}(w)$
- (χρήσιμη προγραμματιστικά)
- Οπου $1_{[x'=f(x,a,w)]}(w)$ ειναι 1 αν $x' = f(x,a,w)$ και μηδέν αλλοιώς
- $2 \rightarrow 1$: $x_{t+1}= w_t$

Δενδρική δομή καταστατικού μοντέλου



πεπερασμένος ορίζοντας

- **Κανόνες απόφασης** της μονάδας: **ντιτερμινιστικοί Μαρκοβιανοί** :
- a_t=$\mu_t(x_t)$, $\mu_t: \mathcal{X} \to \mathcal{A}, \mathrm{t} = 0,1, \dots, \mathrm{T}-1$
- Θεωρητικά επαρκούν για τη περίπτωση μιας υπολογιστικής μονάδας
- Αντίθετα οι **στοχαστικοί κανόνες απόφασης** ειναι:
  - αλγοριθμικά σημαντικοί (βελτιστοποίηση πολιτικής),
  - αναγκαίοι στις πολλαπλές μονάδες και
  - παρέχουν ικανότητα εξερευνησης
- **Πολιτική:** $\pi = (\mu_0, \mu_1, \dots, \mu_{\mathrm{T}-1})$

πεπερασμένος ορίζοντας

- **Συνάρτηση αξίας της πολιτικής π** συνολική ανταμειβή που προκύπτει για κάθε κατάσταση x οταν εφαρμόζεται η π.
- $V_\pi(x) = \mathrm{E}[\beta^T r_T(x_T) + \sum_{t=0}^{T-1} \beta^t r_t(x_t, \mu_t(x_t), w_t) | \, x_0 = x]$

ή ισοδύναμα

- $V_\pi(x) = \mathrm{E}[\beta^T r_T(x_T) + \sum_{t=0}^{T-1} \beta^t r_t(x_t, \mu_t(x_t), x_{t+1}) | \, x_0 = x]$
- r_t **στιγμιαία συνάρτηση ανταμειβής**, και $\beta^T r_T(x_T)$ τελική υπολείπουσα αξία
- **Συνάρτηση αξίας (value function):**
- $V(x) = \max_\pi V_\pi(x)$
- **Βέλτιστη πολιτική** $\pi^* \in arg \max_\pi V_\pi(x)$

Αρχή της αριστότητας

- Αν $\boldsymbol{\pi}^* = (\mu_0, \mu_1, \ldots, \mu_{T-1})$ βέλτιστη πολιτική τότε για κάθε t, η πολιτική $(\mu_t, \mu_{t+1}, \ldots, \mu_{T-1})$ είναι επίσης βέλτιστη για το υποπρόβλημα που ξεκινά την στιγμή t.
- Αν η ταχύτερη διαδρομή απο την Αθήνα στη Θεσσαλονίκη περνά απο τη Λαμία, τοτε το τμήμα απο τη Λαμία στη Θεσσαλονίκη είναι επίσης το ταχύτερο.

Δυναμικός προγραμματισμός

- Αντίστροφη στο χρόνο (backward) αναδρομική σχέση:
- Αρχική συνθήκη
- $V_{\mathrm{T}}(x) = \beta^{T} r_{\mathrm{T}}(x)$, για κάθε x
- Ανάστροφη αναδρομή: Για κάθε χρονική στιγμή t=T-1, T-2,…, 0 :
- $V_t(x) = \max_{a \in \mathcal{A}(x)} \mathrm{E}_w[\beta^t r_t\,(x, a, w) + V_{t+1}\big(f_t(x, a, w)\big)],\ \forall\ x\ \ (*)$
- Η συνάρτηση αξίας προκύπτει ύστερα απο Τ βήματα:
- $V_0(x) = \mathrm{V(x)}$
- Επιπλέον, η πολιτική που μεγιστοποιεί το δεξιό μέλος της (*) καθορίζει τη βέλτιστη πολιτική

Δυναμικός προγραμματισμός

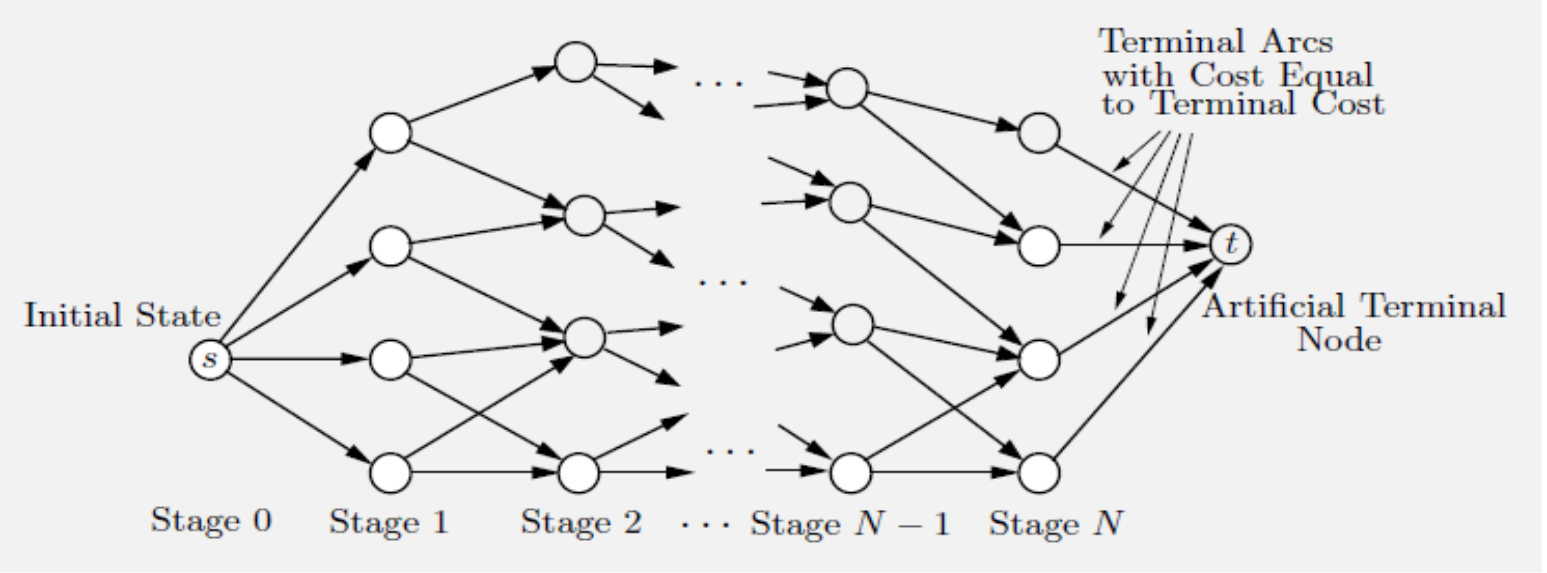
- Γενικές παρατηρήσεις
- Η απόδειξη προκύπτει με επαγωγή που ξεκινά τη τελική χρονική στιγμή T (Berts 95).
- Τεχνικά πιο σύνθετη σε συνεχείς καταστατικούς/ενεργειακούς χωρους.
- Θεωρητικά το πρόβλημα της μεγιστοποίησης ως προς την ενέργεια στο δεξιό μέλος είναι προφανές οταν ο χώρος ενεργειών πεπερασμένος.
- Η επίδοση δεν βελτίωνεται αν χρησιμοποιηθούν στοχαστικές πολιτικές, η πολιτικές που αξιοποιούν όλο το ιστορικό της κατάστασης
- Η ελαχιστοποίηση μετατρέπεται σε μεγιστοποίηση παίρνοντας την αντίθετη της ανταμειβής $r \to -r$
- Η περίπτωση συνεχών ενεργειών με τους περιορισμούς A(x), απαιτεί τεχνικές μη γραμμικού προγραμματισμού
- Κλειστές εκφρασεις για την αξία και την πολιτική έχουμε σε λίγες περιπτώσεις με προεξάρχουσα τη περίπτωση οπου το περιβάλλον είναι γραμμικό και η ανταμειβή έχει τετραγωνική μορφή. Επίσης δομικές πολιτικές (structural policies) με συγκεκριμένα χαρακτηριστικά (μονοτονία, cutoff) αναδύονται σέ προβλήματα ουρών αναμονής, βέλτιστου τερματισμού, διαχείρισης αποθήκης.

Ντιτερμινιστικό περιβάλλον

- Ο αλγόριθμος δυναμικού προγραμματισμού απλοποείται γιατί δεν χρειάζεται η μέση τιμή (απουσία του w)
- $V_t(x) = \max_{a \in \mathcal{A}(x)} [\beta^t r_t(x,a) + V_{t+1}\big(f_t(x,a)\big)], \ \forall\, x$
- Η αλληλεπίδραση με τον πράκτορα και η αναζήτηση της βέλτιστης συμπεριφοράς περιγράφονται ισοδύναμα με την εύρεση βέλτιστου μονοπατιού μεταξύ δύο κόμβων ενός γράφου. Ισχύει και το αντίστροφο:
- Κάθε πρόβλημα ευρεσης βέλτιστου μονοπατιού (κόστος μονοπατιού το άθροισμα του κόστους των ακμών που το αποτελούν) μπορει να περιγραφεί ως ντιτερμινιστικό πρόβλημα δυναμικού προγραμματισμού και να λυθεί με τον αλγόριθμο DP. (Εφαρμογές στην επιχειρησιακή έρευνα,σε προβλήματα δικτύων, Bellman-Ford, Viterbi, κλπ)

Ντιτερμινιστικό περιβάλλον

- Μπορούν συνεπώς να χρησιμοποιηθούν οι αλγόριθμοι RL που θα μελετήσουμε στο μάθημα



Ντιτερμινιστικό περιβάλλον

- Προβλήματα κοντινότερης διαδρομής: μεταφορές, δίκτυα
- Προβλήματα μακρινότερης διαδρομής (διαδρομές με μεγαλύτερη αξία, critical path analysis):
- Ενα project αποτελείται απο έργα (tasks).
- Κάθε έργο αναπαρίσταται με μια ακμή. Το μήκος της δηλώνει το χρόνο ολοκλήρωσης του. Το έργο δεν μπορεί να ολοκληρωθεί αν δεν έχουν ολοκληρωθεί οσα έχουν προηγηθεί.
- μακρινότερη διαδρομή: ελάχιστος χρόνος για την ολοκλήρωση όλων των έργων και συνεπώς του project.

Ντιτερμινιστικό περιβάλλον

- Παράδειγμα:
- Περιπλανώμενος πωλητής
- (travelling salesman)
- Berts2019

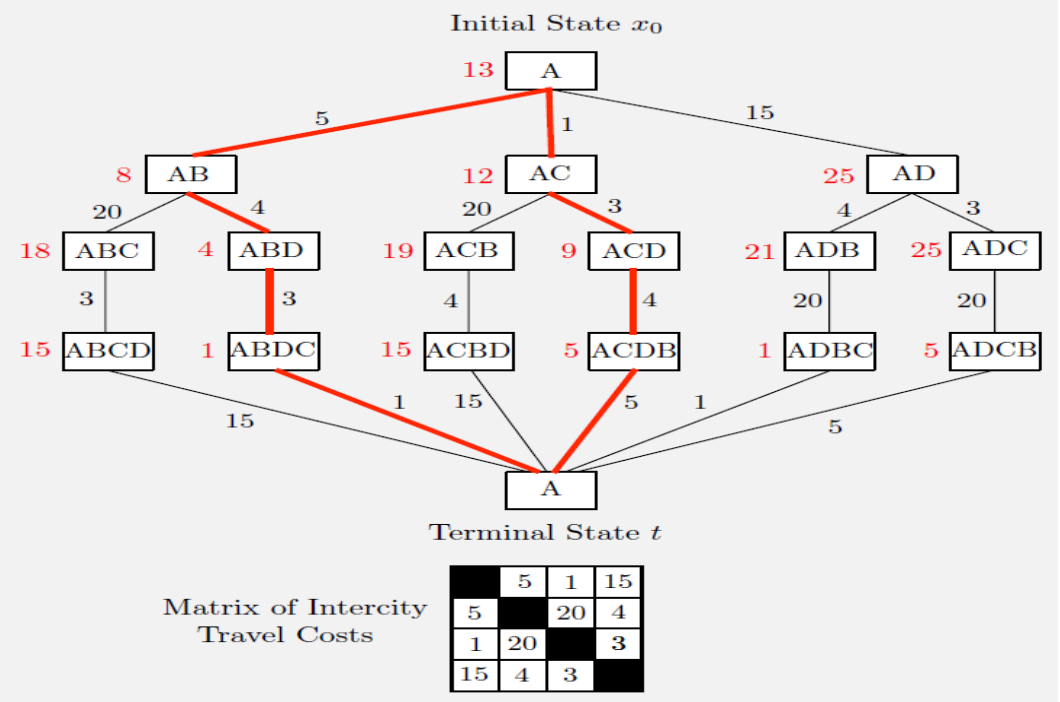


Figure 1.3.2 Example of a DP formulation of the traveling salesman problem. The travel times between the four cities A, B, C, and D are shown in the matrix at the bottom. We form a graph whose nodes are the k-city sequences and correspond to the states of the kth stage. The transition costs/travel times are shown next to the arcs. The optimal costs-to-go are generated by DP starting from the terminal state and going backwards towards the initial state, and are shown next to the nodes. There are two optimal sequences here (ABDCA and ACDBA), and they are marked with thick lines. Both optimal sequences can be obtained by forward minimization [cf. Eq. (1.7)], starting from the initial state x_0.

Διαχείριση εμπορευμάτων αποθήκης (inventory management)

- Σημαντικό πρόβλημα στην εφοδιαστική αλυσίδα.
- Μελετήθηκε αρχικά ως MDP απο διακεκριμμένους οικονομολόγους οπως οι Arrow και Marschak
- Οι δημοσιεύσεις των Arrow, Harris and Marschak 1951 και Dvoretsky Kiefer and Wolfowitz 1952, εκτόξευσαν τη δημοτικότητα των μεθόδων MDP.
- **AMAZON**
- Ιδρύθηκε απο τον Jeff Bezos το 1994 αρχικά ως διαδικτυακό βιβλιοπωλείο.
- Εσοδα το 2004, 7δισ. $.
- Εσοδα το 2016, 136 δισ. $. Ετήσιος ρυθμός ανάπτυξης 20% !!
- Η μεγαλυτερη παγκοσμίως παρουσία στο λιανικό εμπόριο σέ έσοδα και χρηματιστηριακή αξία, δεύτερη σε πωλήσεις μετά την Alibaba.

Διαχείριση εμπορευμάτων αποθήκης

- **Το κλειδί της επιτυχίας**: καινοτόμα και εξαιρετικά αποτελεσματικά logistics στην εφοδιαστική αλυσίδα. Κολοσιαιο δίκτυο απο οργανισμούς και επιχειρηματικές διαδικασίες για:
- Συλλογή πρώτων υλών και μετατροπή σε ενδιάμεσα και τελικά προιόντα
- Διανομή και παράδοση στους πελάτες
- 120 γιγαντιαίες αποθήκες, 45000 ρομποτ (δεδομένα πριν τον κορωνοιό)
- Εξαιρετικό και διαρκώς βελτιούμενο inventory και scheduling management για εγγυημένη παράδοση στον ελάχιστο χρόνο

Διαχείριση εμπορευμάτων αποθήκης

- Ενα είδος εμπόρευματος, μια αποθήκη
- Κατάσταση x_t: εμπορεύματα στην αποθήκη
- Ενέργειες a_t: Παραγγελίες ποσότητας εμπορευμάτων
- Τυχαιότητα w_t: στοχαστική ζήτηση για εμπορεύματα
- Ανταμειβή: περιλαμβάνει το κόστος αποθήκης $r(x_t)$, το κόστος παραγγελίας ca_t εμπορευμάτων και γενικότερα τα κέρδη, τα έσοδα απο τις πωλήσεις μείον το κόστος.
- Υπερβάλλουσα ζήτηση: είτε εξυπηρετείται όταν υπάρξει διαθεσιμότητα με νέες παραλαβές, είτε χάνεται με απώλεια εσόδων
- Ζητήματα και επιλογές στη κατασκευή προτύπων.

Διαχείριση εμπορευμάτων αποθήκης

- Πρώτη εκδοχή (παράδειγμα 1.1.1, Bertsekas 1995)
- $x_{t+1} = x_t + a_t - w_t$ (1)
- Παραγγελίες που δεν ικανοποιούνται απο το τρέχον αποθεματικό, παραμένουν ενεργές μέχρι να ικανοποιηθούν.
- Μοντέλα ανταμειβής: (ι) κόστος (ιι) κέρδη (εσοδα μειο εξόδα)
- (ι) Ανταμειβή=κόστος=κόστος αποθήκης και κόστος παραγγελιών)
  - $r(x_t) + ca_t$ (κόστος αποθήκης και κόστος παραγγελιών)
  - Τετραγωνικό κόστος αποθήκης: $(x_t + a_t - w_t)^2$,
  - οπότε συνολικο κόστος $ca + (x_t + a_t - w_t)^2$
  - Γραμμικό κόστος: $hx_t + ca_t$
  - Προσθήκη σταθερού κόστος παραγγελιών $K1_{a_t>0} + ca_t + hx_t$
- Δεύτερη εκδοχή (παράδειγμα 1.3.2, Bertsekas 1995): Αν η ζήτηση δεν ικανοποιείται, ο πελάτης στρέφεται σε αλλο προιόν.
- $x_{t+1} = \max(0, x_t + a_t - w_t)$ (2)
- Χωρητικότητα αποθήκης M (πχ τόννοι, αυτοκίνητα),
- $x_t + a_t \leq \mathrm{M}$

Διαχείριση εμπορευμάτων αποθήκης

- Τρίτη εκδοχή (M. Puterman, 2005, section 3.2, C. Szepesvari, 2010 παράδειγμα 1.1)
- Διαχείριση αποθήκης με απώλειες εσόδων και μεγιστοποίηση κερδών.
- Αποθήκη μέγιστης χωρητικότητας M και αβέβαιης ζήτησης.
- Απόγευμα ημέρας t: Ο διαχειριστής παρατηρεί το εμπόρευμα στην αποθήκη και αποφασίζει τις παραγγελίες της επόμενης ημέρας
- Πρωί ημέρας t+1: Παραλαβή παραγγελιών
- Κατά τη διάρκεια της ημέρας t+1 εκδηλώνεται στοχαστικά η ζήτηση
- Χωρος καταστάσεων και ενεργειών: {0,1,2,...,M}
- Η κατάσταση στην αποθήκη x_t απο το απόγευμα της ημέρας t, μέχρι το απόγευμα της επόμενης ημέρας:
- $x_{t+1} = \max(0, \min(x_t + a_t, M) - w_{t+1}))$ (απαλλαγή απο τον περιορισμό) (3)

Διαχείριση εμπορευμάτων αποθήκης

- Τρίτη εκδοχή: οπως το (2) αλλά
- Ανταμειβή=κέρδος=Εσοδα μείον έξοδα
- Εξόδα: Σταθερό κόστος εκκίνησης της παραγγελίας, γραμμικό κόστος παραγγελίας, γραμμικό κόστος αποθήκης.
- Εσοδα: $p * q$, q οι πωλήσεις και p η τιμή μονάδας.
- Γραμμικά εσοδα: $p(x_t + a_t - x_{t+1})$
- Παράδειγμα ανταμειβής κέρδους
- $p(x_t + a_t - x_{t+1}) - ca_t - hx_t$ η ακόμα
- $r_{t+1}(x_t, a_t, x_{t+1}) = -K1_{a_t>0} - ca_t - hx_t + pmax\{0, \min(x_t + a_t, M\} - x_{t+1}\}$
- Σημείωση (Αυτή η εκδοχή εμφανίζεται στο third party envs, OpenAI gym. Καποια σημεία χρειάζονται αποσαφήνιση)
- Γενικεύσεις: παραδόσεις με υστέρηση, ζήτηση με χρονική συσχέτιση, πολλά προιόντα και τύποι.

Διαχείριση εμπορευμάτων αποθήκης

- Παράδειγμα, Εκδοχή 2. (Bertsekas 95).
- Χρονικός ορίζοντας, T=3
- Η ζήτηση διακρίνεται σε χαμηλή, μέτρια και υψηλή με πιθανότητες
- $p(0) = .1, \;\; p(1) = .7, \;\; p(2) = .2$
- Τελική υπολείπουσα ‘αξία’ $V_3(x) = 0$
- Αρχικό εμπόρευμα στην αποθήκη $x_0 = 0$
- **Δυναμικός προγραμματισμός, περίοδος T-1=2**
- Υπολογισμός της συνάρτησης αξίας $V_2(x)$: $V_2(0)$, $V_2(1)$, $V_2(2)$
- Επειδή $V_3(x) = 0$, για κάθε x, ο δυναμικός προγραμματισμός δίνει:

Διαχείριση εμπορευμάτων αποθήκης

- $V_2(x) = min_{0 \le a \le 2-x} E_w[a + (x + a - w)^2]$
- Υπολογίζω την $V_2(x)$ για κάθε x
- $V_2(0) = min_{0 \le a \le 2} E_w[a + (a - w)]^2 = min_{0 \le a \le 2}[a + 0.1a^2 + 0.7(a-1)^2 + 0.2(a-2)^2]$
- Βρίσκω το ελάχιστο
- $V_2(0) = \min\{1.5, 1.3, 3.1\} = 1.3$
- Επιτυγχάνεται με την ενέργεια a=1. Αρα
- $V_2(0) = 1.3, \ \ \mu_2(0) = 1$

Διαχείριση εμπορευμάτων αποθήκης

- Με τον ίδιο τρόπο βρίσκω την αξία στη κατάσταση x=1 τη περίοδο 2, οπου οι επιτρεπτές ενέργειες είναι οι a=0, 1.
- $V_2(1) = min_{a=0,1} E_w[a + (a + 1 - w)]^2 = min_{a=0,1}[a + 0.1(a+1)^2 + 0.7(a)^2 + 0.2(a-1)^2$
- $V_2(1) = \min\{0.3, 2.1\} = 0.3, \quad \mu_2(1) = 0$
- Για x=2, η μόνη επιτρεπτή ενέργεια είναι a=0, οπότε
- $V_2(2) = 1.1, \quad \mu_2(2) = 0$

Διαχείριση εμπορευμάτων αποθήκης

- **Περίοδος Τ-2=1.**
- $V_1(x) = min_{0 \le a \le 2-x} E_w[a + (x + a - w)^2 + V_2(\max(0, x + a - w))]$
- x=0
- $V_1(0) = min_{a=0,1,2} E_w[a + (a - w)^2 + V_2(\max(0, a - w))]$
- Για a=0, η μέση τιμή είναι
- $0.1V_2(0) + 0.7(1 + V_2(0) + 0.2\big(4 + V_2(0)\big) = 2.8$
- Για a=1, η μέση τιμή είναι 2.5
- Για a=2, η μέση τιμή είναι 3.68
- Αρα $V_1(0)$=2.5, $\mu_1(0) = 1$

Διαχείριση εμπορευμάτων αποθήκης

- Συνεχίζοντας με τον ίδιο τρόπο καταλήγουμε στον πίνακα: (Αναλυτικοί υπολογισμοί D. Bertsekas 95vol. I pages 23-27.
- Η βέλτιστη πολιτική είναι φθίνουσα συνάρτηση της κατάστασης και σταθερή στο χρόνο (στάσιμη).

Stock	Stage 0 Cost-to-go	Stage 0 Optimal stock to purchase	Stage 1 Cost-to-go	Stage 1 Optimal stock to purchase	Stage 2 Cost-to-go	Stage 2 Optimal stock to purchase
0	3.7	1	2.5	1	1.3	1
1	2.7	0	1.5	0	0.3	0
2	2.818	0	1.68	0	1.1	0

Διαχείριση εμπορευμάτων αποθήκης: υλοποίηση

- Υλοποίηση με τη χρήση της βιβλιοθήκης quantecon:
- Advanced quantitative economics with Python, T. Sargent and J. Stachurski, Sept. 2020
- Καλύπτει υποδείγματα απο τη θεωρία παιγνίων, τις αλυσίδες Markov, τον δυναμικό προγραμματισμό.
- Μέθοδοι DP που βασίζονται στη γνώση του μοντέλου και σε πεπερασμένα σύνολα καταστάσεων και ενεργειών.
- Η κλάση DiscreteDP κατασκευάζει περιβάλλοντα στα οποία δίνονται (i) ο πίνακας ανταμειβής R(s,a) και ο τανυστής των πιθανοτήτων μετάβασης P(s'|s,a) (χρησιμοποιείται το s αντι του x για τη κατάσταση)

Διαχείριση εμπορευμάτων αποθήκης: υλοποίηση

- Υλοποιεί τους αλγορίθμους:

1. Δυναμικός προγραμματισμός, πεπερασμένος ορίζοντας
2. Επανάληψη αξίας (value iteration) άπειρος ορίζοντας
3. Επανάληψη πολιτικής (policy iteration) άπειρος ορίζοντας
4. Τροποποιημένη επανάληψη πολιτικής (modified) άπειρος ορίζοντας
5. Γραμμικός προγραμματισμός (linear programming)

- Οι μέθοδοι 2-5 θα μελετηθούν αργότερα στο μάθημα 2.

Διαχείριση εμπορευμάτων αποθήκης: υλοποίηση

- Για την περιγραφή των αντικειμένων της κλάσης:
- quantecon.markov.ddp.DiscreteDP(R, Q, beta) χρησιμοποιούνται δύο τρόποι ορισμού των παραμέτρων R, Q:

1. DiscreteDP(R, Q, beta)

- $R, n \times m$ συστοιχία (2D array) ανταμειβής
- Q, $n \times m \times n$ τανυστής (3D array) με τις πιθανότητες μετάβασης
- Προσοχή στην αλλαγή συμβολισμού (Q αντι P, s αντι x)
- beta συντελεστής προεξόφλησης
- https://python-advanced.quantecon.org/discrete_dp.html

Διαχείριση εμπορευμάτων αποθήκης: υλοποίηση

2. DiscreteDP(R, Q, beta, s_indices, a_indices)

- R διάνυσμα (1D array) μήκους L
- Q πίνακας (2D array) μήκους $L \times n$
- s_indices, a_indices διανύσματα μήκους L
- Το μήκος L καταγράφει το πλήθος των επιτρεπτών ζευγαριών καταστάσεων και ενεργειών.
- Για i=0,1,2,…,L-1, κάθε ζευγάρι [s_indices[i], a_indices[i]] ορίζει επιτρεπτή ενέργεια a_indices[i] στη κατάσταση s_indices[i].
- R[i], η ανταμειβή στο επιτρεπτό ζευγάρι [s_indices[i], a_indices[i]]

Διαχείριση εμπορευμάτων αποθήκης: υλοποίηση

- Q[i, s_next] πιθανότητα η επόμενη κατάσταση να είναι s_next, όταν η τρέχουσα κατάσταση είναι s_indices[i] και η ενέργεια a_indices[i].
- Η περίπτωση πεπερασμένου χρονικού ορίζοντα και η ανάστροφη αναδρομική σχέση του δυναμικού προγραμματισμού υλοποιείται με:
- quantecon.markov.ddp.backward_induction(ddp, T, v_term=None)
- Το αντικείμενο ddp εχει ήδη δημιουργηθεί ως αντικείμενο της κλάσης DiscreteDP (με παραμέτρους R, Q, beta οπου καλό είναι για το παράδειγμα να τεθεί beta=1, ωστε να επαληθευτούν τα παραδείγματα απο τα βιβλία)
- T χρονικός ορίζοντας
- v_term η τελική συνάρτηση αξίας τη στιγμή T, v_term=None δηλώνει οτι είναι μηδέν.

Διαχείριση εμπορευμάτων αποθήκης: υλοποίηση

- Επιστρέφει τη βέλτιστη αξία vs και τη βέλτιστη πολιτική sigmas σε κάθε χρονική στιγμή t=0,1,2,… T-1.
- vs: ndarray(float, ndim=2)
- Array of shape (T+1,n) οπου vs[t] η βέλτιστη συνάρτηση αξίας τη στιγμή t ως διάνυσμα μήκους n (όσες είναι οι καταστάσεις), t=0:T
- sigmas: ndarray(int,ndim=2)
- Array of shape (T,n) οπου sigmas[t] η βέλτιστη συνάρτηση πολιτικής τη στιγμή t ως διάνυσμα μήκους n, t=0:T-1.
- Επίδειξη στο Spyder

Διαχείριση εμπορευμάτων αποθήκης: υλοποίηση

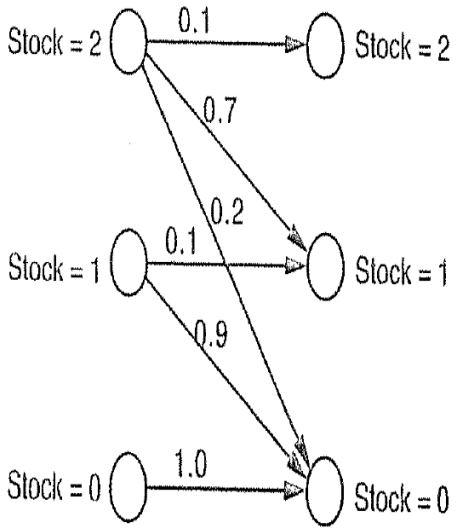
- res
- Out[90]:
- (array([[-3.7 , -2.7 , -2.818],
- [-2.5 , -1.5 , -1.68],
- [-1.3 , -0.3 , -1.1],
- [0.   , 0.   , 0.   ]]),
- array([[1, 0, 0],
- [1, 0, 0],
- [1, 0, 0]]))

Διαχείριση εμπορευμάτων αποθήκης: υλοποίηση

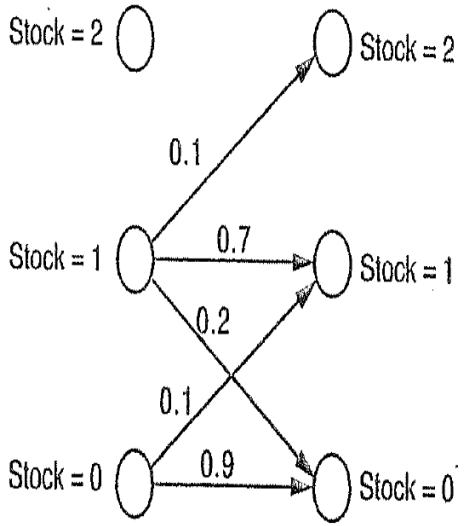
- a_indices
- Out[30]: [0, 1, 2, 0, 1, 0]
- s_indices
- Out[31]: [0, 0, 0, 1, 1, 2]
- R
- Out[32]:
- [-1.5,
- -1.3,
- -3.099999999999996,
- -0.3000000000000004,
- -2.099999999999996,
- -1.1]

Διαχείριση εμπορευμάτων αποθήκης: υλοποίηση

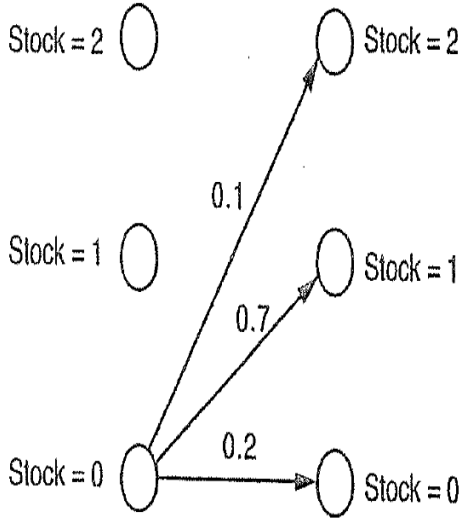
- Q
- Out[101]:
- [array([1., 0., 0.]),
- array([0.9, 0.1, 0.]),
- array([0.2, 0.7, 0.1]),
- array([0.9, 0.1, 0.]),
- array([0.2, 0.7, 0.1]),
- array([0.2, 0.7, 0.1])]



Stock purchased = 0



Stock purchased = 1



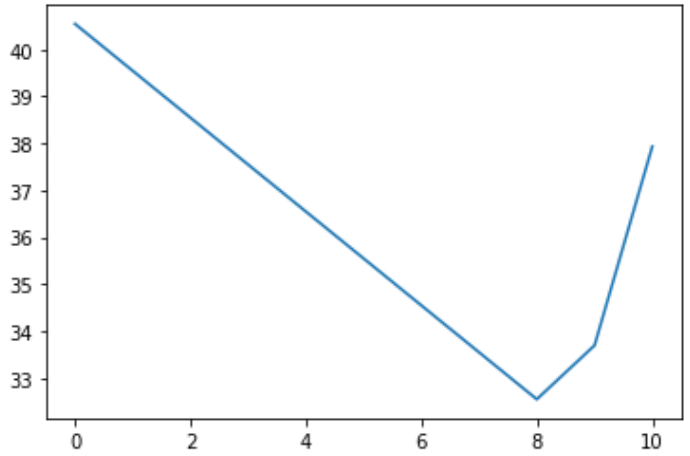
Stock purchased = 2

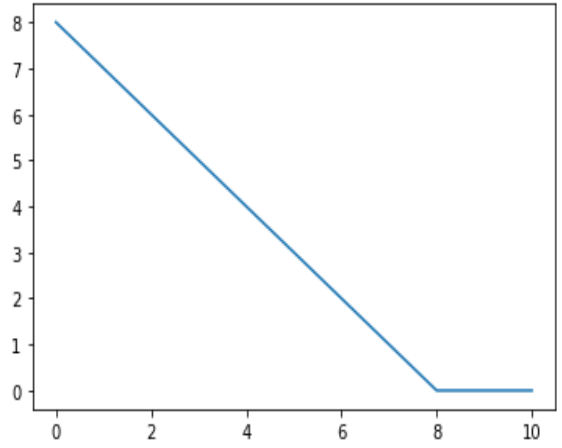
Διαχείριση εμπορευμάτων αποθήκης: υλοποίηση

- Τι συμβαίνει όταν ο χρονικός ορίζοντας μεγαλώνει?
- Επίδειξη στο spyder
- Εστιάζω στη βέλτιστη αξία και στη βέλτιστη πολιτική
- T=10, vs[0], sigmas[0]

Διαχείριση εμπορευμάτων αποθήκης: υλοποίηση

- Γενικότερη εικόνα: αυθαίρετες τιμές για τη χωρητικότητα, μεγαλύτερη ετερογένεια στη ζήτηση, χρονικό ορίζοντα:
- Πειραματική Ανάλυση ευαισθησίας της βέλτιστης αξίας vs[0] και της βέλτιστης πολιτικής ως προς τις παραμέτρους:
- Cap, W, p_w, T
- Επίδειξη στο Spyder
- Δομημένες πολιτικές





NumPy

- Η βιβλιοθήκη quantecon στηρίζεται στη Numpy, βιβλιοθήκη χειρισμού πολυδιάστατων συστοιχιών και αντίστοιχων δομών δεδομένων (multidimensional arrays ή ndarrays).
- Ακολουθεί σύντομη εισαγωγή εντολών που χρειάζονται στην υλοποίηση
- πηγή: NumPY: user guide, NumPy: the absolute basics for beginners)
- Βασικό αντικείμενο της κλάσης **ndarray**, το N-διάστατο array: $x(n_1, n_2, \cdots, n_N)$
- Η διάσταση N, οποιοσδήποτε ακέραιος. Οι μεταβλητές n_i κυμαίνονται στα διαστήματα $0: N_i$
- Διάνυσμα N=1, πίνακας N=2, τανυστής N=3.

NumPy

- Κάθε ndarray είναι συλλογή τεμαχίων ίδιου τύπου και μεγέθους
- import numpy as np
- x=np.array([[1,2,3],[4,5,6]]) # έχει δύο αξόνες (axis),
- # ο πρώτος έχει μήκος 2 και ο δεύτερος μήκος 3
- # Ορισμένα βασικά χαρακτηριστικά (attributes)
- x.dtype # τύπος δεδομένων dtype('int32')
- x.shape # διανυσμα ακεραίων που δείχνουν τον αριθμό των στοιχείων σε καθε αξονα
- # Τι σημαίνει x.shape=(3,2,4)? Η συλλογή αποτελείται απο 3 τεμάχια
- # Καθε τεμάχιο έχει 2 τεμάχια και καθε ένα απο αυτά 4 στοιχεία
- x.ndim # αριθμός αξόνων (διάσταση)
- x.size #συνολικό πλήθος στοιχείων

NumPy: συντομα εισαγωγικά στοιχεία

- # Δημιουργία συστοιχιών
- np.array() # εισαγωγή στοιχείων πχ
- a=np.array([1,2,3]) # προσοχή στη γραφή ([,])
- np.zeros(3) # προσοχή ο τυπος ειναι 'float'
- np.ones(3) # ομοίως. Για αλλαγή σε ακέραιους:
- np.ones(3,dtype=np.int64)
- np.empty(4) # εισάγει τυχαία στοιχεία ανάλογα με τη κατάσταση της μνήμης. Δεν ξεχνάμε να το συμπληρώσουμε αργότερα
- np.arange(4)   # διάστημα συνεχόμενων ακεραίων με αρχή το 0 και τέλος 3

NumPy

- np.arange(n,m,k) # διάστημα ισαπεχόντων ακεραίων με αρχή το n, τέλος m
- # (δεν συμπεριλαμβάνεται) και απόσταση μεταξύ τους k
- np.arange(2,9,2) # array([2, 4, 6, 8])
- np.linspace(n,m,k) # Διαμέριση διαστήματος με αρχή το n τέλος m σε k ίσα τμήματα
- x=np.linspace(0,2*np.pi,100) # 100 ισαπέχοντα σημεία στο διαστημα [0,2π]
- y=np.sin(x)
- import matplotlib.pyplot as plt
- plt.plot(x,y)
- b=np.zeros.like(x) # έχω ήδη array x και δημιουργώ το y αποτελούμενο
- # απο μηδενικά αλλά με τα χαρακτηριστικά του χ

NumPy

- np.fromfunction # κατασκευάζει array υπολογίζοντας μια συνάρτηση σε κάθε διάνυσμα συντεταγμένων
- np.fromfunction(lambda i,j: i+j, (3,3),dtype=int) # (3,3) shape
- # Out[20]:
- array([[0, 1, 2],
- [1, 2, 3],
- [2, 3, 4]])

NumPy

- # Τεμαχισμός (slicing) και Δεικτοδότηση (indexing)
- d=np.array([1,2,3])
- d[1]
- d[0:2]
- d[1:] # απο το δεύτερο στοιχέιο μέχρι τέλος
- y=np.arange(35).reshape(5,7)

NumPy

- # Ενα array μπορεί να δεικτοδοτηθεί απο άλλο array (index array) (τύπος ακέραιοι)
- # Το index array μπορεί να έχει διάσταση > 1.
- x=np.arange(10,1,-1)
- x[np.array([3,3,1,8])]
- x[np.array([[1,1],[2,3]])]

Διαχείριση εμπορευμάτων αποθήκης: υλοποίηση

- Κρίσιμες παράμετροι ‘κλιμακας’

1. Χωρητικότητα αποθήκης, cap
2. Πλήθος τιμών ζήτησης, n_w
3. Χρονικός ορίζοντας, T

- Το πολύ απλό παράδειγμα είναι η εξαίρεση και όχι ο κανόνας. Στη πραγματικότητα πολλές αποθήκες πολλά προιόντα πολλοί τύποι υστερήσεις στη παράδοση κλπ καθιστούν τη περιγραφή των R και Q δύσκολη και την πολυπλοκότητα του αλγορίθμου μεγάλη.
- Στις πραγματικές εφαρμογές (ρομποτικη παιγνίδια,..) οι χώροι καταστάσεων και ενεργειών είναι πολύ μεγάλοι και σύνθετοι
- Για μεγάλο χρονικό ορίζοντα προσεγγίζουμε $T=\infty$ οποτε προκύπτει μια σημαντική απλοποίηση: η βέλτιστη πολιτική είναι στάσιμη (επόμενες διαφάνειες)

Απειρος χρονικός ορίζοντας

- Ο νόμος της κίνησης και η συνάρτηση ανταμειβής δεν αλλάζουν στο χρόνο.
- Θα αγνοήσουμε τεχνικά σημεία οπως η εναλλαγή μέσης τιμής και ορίου.
- Εξετάζουμε τι συμβαίνει στο δυναμικό προγραμματισμό όταν ο ορίζοντας μεγαλώνει, $\mathrm{T} \to \infty$
- Αρχική συνθήκη
- $V_{\mathrm{T}}(x) = \beta^{T} \mathrm{V}_0(x)$, $\mathrm{V}_0(x)$ οποιαδήποτε (φραγμένη) συνάρτηση.
- $V_{\mathrm{T}-t}(x) = \max_{a \in \mathcal{A}(x)} E[\beta^{T-t} r\ (x, a, w) + V_{\mathrm{T}-t+1}\big(f(x, a, w)\big)]$, t=1,2,…,T-1
- Η εξαγωγή του β^{T-t} έξω απο το max και η $V_{\mathrm{T}-t}(x)/\beta^{T-t} \leftarrow \mathrm{V}_t(x)$
- οδηγεί στην αναδρομή

$$V_{t+1}(x) = \max_{a \in \mathcal{A}(x)} E[r\ (x, a, w) + \beta V_t\big(f(x, a, w)\big)] \qquad (**)$$

Επισκόπηση βασικών αποτελεσμάτων

1. Εστω μηδενική υπολείπουσα αξία: $V_0(x) = 0$. Λογικό να εικάσουμε οτι όταν $T \to \infty$ το όριο στη σχέση (**) αν υπάρχει, ικανοποιεί την

- **εξίσωση Bellman**
- Οταν η καταστατική ανανέωση με τον νόμο της κίνησης:
- $V(x) = \max_{a \in \mathcal{A}(x)} E[r\,(x, a, w) + \beta V\big(f(x, a, w)\big)]$ (BE)
- ή
- $V(x) = \max_{a \in \mathcal{A}(x)} \sum_{x'} P(x'|x, a)\,[r\,(x, a, x') + \beta V(x')]$
- Εμμεση (implicit) μη γραμμική αλγεβρική εξίσωση.

Επισκόπηση βασικών αποτελεσμάτων

- Για πεπερασμένες καταστάσεις, η (BE) αποτελεί ένα σύστημα μη γραμμικών εξισώσεων.
- 2. Τα παραπάνω ισχύουν για κάθε φραγμένη αρχική συνάρτηση γιατί

$$\beta^T \mathrm{V}_0(x) \to 0$$

- 3. Αν ο κανόνας απόφασης $\mu(x)$ επιτυγχάνει το μέγιστο στο δεξιό μέλος της (BE):
- $\mu(x)=\underset{a\in\mathcal{A}}{\operatorname{argmax}} \sum_{x'} P(x'|x,a)\,[r\ (x,a,x') + \beta V(x')] \equiv greedy(V)(x)$
- τότε η **στάσιμη** πολιτική $\pi = (\mu, \mu, \dots)$ ειναι βέλτιστη

Σταθερό σημείο

- Δύο διαδεδομένοι τρόποι επίλυσης της εξ. Bellman είναι:
- 1. προσδιορισμός σταθερού σημείου
- T(V)=V
- Με τον αλγόριθμο σταθερού σημείου
- V_{t+1} =T(V_t)
- όπου
- T(V)(x)= $\max_{a \in \mathcal{A}(x)} \sum_{x'} P(x'|x,a)\,[r\ (x,a,x') + \beta V(x')]$

Σταθερό σημείο

- Αποδεικνύεται οτι η Τ είναι απεικόνιση συστολής (contraction map)
- $\| T(V) - T(V') \| < L \| V - V' \|$, 0<L<1
- με L=β. Οπότε (απο το θεώρημα Banach) έχει μοναδικό σταθερό σημείο V (βέλτιστη αξία) και ο αλγόριθμος σταθερού σημείου συγκλίνει σε αυτό με ταχύτητα που καθορίζεται απο το β.
- Ο αλγόριθμος σταθερού σημείου
- $\mathrm{V}_{t+1}(x) = max_a \sum_{x'} P(x'|x,a) \left[r\ (x,a,x') + \beta V_t(x')\right]$ (VI)
- ή
- $\mathrm{V}_{t+1}(\mathrm{x}) = \max_{a \in \mathcal{A}(x)} E[r\ (x,a,w) + \beta V_t\big(f(x,a,w)\big)]$

Αλγόριθμος επανάληψης αξίας

- Ονομάζεται **επανάληψη αξίας (value iteration).**
- Η επανάληψη αξίας δίνει στο όριο την (βέλτιστη) αξία V και η βέλτιστη πολιτική υπολογίζεται απο την
- $\mu(x) = \text{argmax}_{a \in \mathcal{A}} \sum_{x'} P(x'|x,a)\,[r\ (x,a,x') + \beta V(x')] \equiv greedy(V)$(x)
- 2. Ενας δεύτερος τρόπος υπολογισμού του σταθερού σημείου είναι με τη μέθοδο Gauss και τη μετατροπή σε πρόβλημα ελαχιστοποίησης. Εστω
- G(V)=T(V)-V
- V είναι σταθερό σημείο της T αν είναι ρίζα (σημείο μηδενισμού) της G. Αρα μπορεί να βρεθεί με τη λύση του
- $\min_{V} \| G(V) \|^2$
- πρόβλημα μη γραμμικού προγραμματισμού χωρίς περιορισμούς

Λύση με γραμμικό προγραμματισμό

- Απο την εξίσωση Bellman προκύπτει οτι η (βέλτιστη) συνάρτηση αξίας ικανοποιεί $|\mathcal{A}||\mathcal{X}|$ γραμμικές ανισότητες
- $V(x) \geq r(x,a) + \beta \sum_{x'} p(x'|x,a)V(x') \quad \forall\, x \in \mathcal{X}, \mathrm{a} \in \mathcal{A}$ (LI)
- Αντίστροφα, κάθε διάνυσμα v(x) που ικανοποιεί την (LI), είναι άνω φράγμα της αξίας:
- v(x) ≥ V(x)
- Πράγματι έστω $\mu(x)$ η βέλτιστη πολιτική. Τότε
- $v(x) \geq r\big(x,\mu(x)\big) + \beta \sum_{x'} p(x'|x,\mu(x))v(x') = V(x)$

Λύση με γραμμικό προγραμματισμό

- Αρα είναι φυσικό να αναζητήσουμε το ‘μικρότερο’ v που ικανοποιεί την (LI).
- Καταλήγουμε σε πρόβλημα γραμμικού προγραμματισμού (LP):
- $\min_{v} \sum_x q(x)v(x)$
- subject to
- $v(x) \geq r(x,a) + \beta \sum_{x'} p(x'|x,a)v(x'), x \in \mathcal{X}, \text{a} \in \mathcal{A}$
- Τα βάρη q(x) είναι αυθαίρετα, αρκεί να είναι θετικά και να αθροίζουν στη μονάδα, π.χ q(x)=1/ $|\mathcal{X}|$ (μπορούν να ερμηνευθούν ως πιθανότητες εμφάνισης των καταστάσεων).

Γραμμικός προγραμματισμός

- Ελαχιστοποίηση/Μεγιστοποίηση γραμμικής συναρτησης με γραμμικούς ανισοτικούς και ισοτικούς περιορισμούς
- $\min_{x} c'x$
- subject to
- $a_i'x \geq b_i$
- $a_i'x \leq b_i$
- $a_i'x = b_i$
- $x_i \geq 0$
- $x_i \leq 0$

Γραμμικός προγραμματισμός

- Πιο οικονομική αλλά ισοδύναμη περιγραφή
- $\min_x c'x$
- subject to
- $a_i'x \geq b_i$
- και σε μορφή πινάκων
- $\min_x c'x$
- subject to
- $\mathrm{A}x \geq \mathrm{b}$

Δυικότητα: Συνέπεια της μεθόδου των πολλαπλασιαστών Lagrange

- **Πρωτεύον πρόβλημα**
- $\min_x c'x$
- subject to
- $\mathrm{A}x \geq \mathrm{b}$
- Πρωτεύον για τη συνάρτηση αξίας
- $\min_v \sum_x q(x)v(x)$
- subject to
- $v(x) \geq r(x,a) + \beta \sum_{x'} p(x'|x,a)v(x'), x \in \mathcal{X}, \mathrm{a} \in \mathcal{A}$

- **Δυικό πρόβλημα**
- $\max_z z'b$
- subject to
- $\mathrm{z}'\mathrm{A} = \mathrm{b}, z \geq 0$
- Δυικό για τη συνάρτηση αξίας

$\max_z \sum_{x,a} r(x,a)z(x,a)$:

$\sum_{x,a}[\delta(x,x') -$

Αξία και βέλτιστη πολιτική ως λύσεις του LP

- Ισχύουν τα εξής

1. Το πρωτεύον και το δυικό πρόβλημα έχουν βέλτιστες λύσεις.
2. Η βέλτιστη λύση του πρωτεύοντος δίνει τη βέλτιστη συνάρτηση αξίας (μοναδική)
3. Εστω z(x,a) βέλτιστη λύση του δυικού προβλήματος. Τότε μετά την κανονικοποίηση, η **στοχαστική** πολιτική

$$\mu(x,a) = \frac{z(x,a)}{\sum_a z(x,a)}$$

συνιστά **βέλτιστη πολιτική**.

Υπολογισμός αξίας συγκεκριμένης πολιτικής

- Διανυσματική περιγραφή
- Εστω π=(μ,μ,...,) δοθείσα στάσιμη πολιτική. Χρησιμοποιούμε το συμβολισμό π και μ ισοδύναμα
- Η στιγμιαία ανταμειβή αν μ ντιτερμινιστική:
- $r\big(x, \mu(x)\big) \equiv r_\pi(x)$
- r_π διάνυσμα με συνιστώσες $r_\pi(x)$: διάνυσμα ανταμειβής
- Ανάλογα ισχύουν αν μ στοχαστική πολιτική:
- $r_\pi(x) = \sum_a r(x,a)\mu(a|x)$
- $P_\pi(x'|x) \equiv P(x'|x, \mu(x))$
- P_π πίνακας με στοιχεία $P_\pi(x'|x)$: πίνακας μετάβασης της αλυσίδας
- Με επαγωγή προκύπτει οτι σε t βήματα
- $P_\pi(x_{t+1} = x' \ |x_0 = x) = P_\pi^t$,
- Υπολογισμός μέσης τιμής μιάς συνάρτησης v(x_t), οταν $x_0 = x$
- $E[v(x_t)|x_0 = x] = \sum_{x'} P_\pi(x_t = x'|x_0 = x)v(x') = P_{\pi,x}^{t-1}v$
- Οπου $P_{\pi,x}^{t-1}$ είναι η x- γραμμή του P_π^t

Υπολογισμός αξίας μιας πολιτικής

- Αξία της πολιτικής π:
- $V_\pi(x) = \mathrm{E}[\sum_{t=0} \beta^T r_t(x_t, \mu_t(x_t), x_{t+1}) | \, x_0 = x]$
- ή
- $V_\pi(x) = E[r_\pi(x, \pi(x), x') + \beta \mathrm{V}_\pi(x')]$ η σε μορφή πίνακα
- $V_\pi = r_\pi + \beta P_\pi \mathrm{V}_\pi = r_\pi + \beta P_\pi r_\pi + \beta {P_\pi}^2 r_\pi + \cdots$
- $V_\pi = (I - \beta P_\pi)^{-1} r_\pi$
- Ο πίνακας $I - \beta P_\pi$ ειναι αντιστρεψιμος, γιατί ιδιοτιμές 1-βλ, |λ|<=1.

Υπολογισμός αξίας μιας πολιτικής

- Αρα ο υπολογισμός της V_π μπορει να γίνει με την επίλυση συστήματος γραμμικών εξισώσεων
- $(I - \beta P_\pi)\mathrm{V} = r_\pi.$
- **Εναλλακτικά με επαναληπτικη μέθοδο σταθερού σημείου, παρατηρώντας οτι η V_π είναι σταθερό σημείο για την απεικόνιση**
- $V \to r_\pi + \beta P_\pi \mathrm{V}$ η οποία είναι συστολή

Επανάληψη πολιτικής

- Δίνεται πολιτική μ_t στο βήμα t:
- **Βήμα 1. Αποτίμηση πολιτικής (policy evaluation)**.
- Υπολόγισε την αξία της πολιτικής μ_t, V_{μ_t} απο τη λύση του γραμμικού συστήματος
- $(I - \beta P_\pi)\mathrm{V} = r_\pi$.
- **Βήμα 2. Βελτίωση πολιτικής (policy improvement).**
- Υπολόγισε νέα πολιτική απο την σχέση (άπληστη πολιτική για την αξία,V_{μ_t})
- $\mu_{t+1}(x) \in arg\max_a r(x,a) + \beta \sum_{x'} P(x'|x,a)V_{\mu_t}(x')$
- Επανέλαβε μέχρι , $V_{\mu_{t+1}} = V_{\mu_t}$

Επανάληψη πολιτικής

- Βηματική βελτίωση: Ισχύει
- $V_{\mu_{t+1}} \geq V_{\mu_t}$
- Πράγματι απο τον ορισμό της μ_{t+1}:
- $r_{\mu_{t+1}} + \beta P_{\mu_{t+1}} V_{\mu_t} \geq r_{\mu_t} + \beta P_{\mu_t} V_{\mu_t} = V_{\mu_t}$
- Άρα
- $r_{\mu_{t+1}} \geq \left(\mathrm{I} - \beta P_{\mu_{t+1}}\right) V_{\mu_t}$ ή
- $\left(\mathrm{I} - \beta P_{\mu_{t+1}}\right)^{-1} r_{\mu_{t+1}} = V_{\mu_{t+1}} \geq V_{\mu_t}$ (επειδή $\left(\mathrm{I} - \beta P_{\mu_{t+1}}\right)^{-1} = \sum \beta^t P^t \geq 0$
- Για πεπερασμένα σύνολα καταστάσεων και ενεργειών ο αλγόριθμος τερματίζει εγγυημένα σε πεπερασμένο αριθμό βημάτων.

Τροποποιημένες και ασύγχρονες εκδοχές

- **Τροποποιημένη επανάληψη πολιτικής (modified policy iteration)**
- Το βήμα της αποτιμήσης πολιτικής εκτελείται με τον αλγόριθμο σταθερού σημείου και μάλιστα με τη χρήση πεπερασμένου αριθμού επαναλήψεων.
- **Επανάληψη αξίας με τη μέθοδο Gauss Seidel:**
- Η αξία στη κατάσταση i στο βήμα t+1 χρησιμοποιεί τις αξίες του βήματος t+1 για τις καταστάσεις 1,2,..., i-1 (αντί αυτών του βήματος t)
- **Ασύγχρονη επανάληψη αξίας**
- Σε κάθε βήμα t+1 επιλέγεται μία κατάσταση, έστω i, ενημερώνεται η αξία της κατάστασης i ενώ οι αξίες των άλλων καταστάσεων παραμένουν οι ίδιες.
- **Ασύγχρονη επανάληψη πολιτικής**: παρόμοια προσαρμογή των παραπάνω