ΣΤΟΧΑΣΤΙΚΕΣ ΑΝΕΛΙΞΕΙΣ

ΕΡΓΑΣΤΗΡΙΟ ΙΙΙ

Άσκηση 1 Στην άσκηση αυτή γίνεται προσομοίωση της αλυσίδας της άσκησης 6 του Φυλλαδίου IV, για $p = 1/2$. Υπενθυμίζεται ότι αυτή η αλυσίδα στον χώρο καταστάσεων $\mathbb{N} \cup \{0\}$ είναι μη υποβιβάσιμη, γνησίως επαναληπτική, με αναλλοίωτη κατανομή π που δίνεται από την

$$\pi(k) = \frac{1}{2^{k+1}}, \quad k = 0, 1, 2, 3, \ldots$$

Κατεβάστε και τρέξτε τον κώδικα `ergodic.py`. Ο κώδικας αυτός προσομοιώνει τα πρώτα $N = 10^6$ βήματα της αλυσίδας και επιστρέφει τον εργοδικό μέσο

$$\frac{X_1 + X_2 + \cdots + X_N}{N}.$$

Από το εργοδικό θεώρημα η παραπάνω ποσότητα συγκλίνει καθώς $N \to \infty$ με πιθανότητα 1 στο

$$\sum_{k=0}^{\infty} k\pi(k),$$

οπότε ο κώδικας υπολογίζει αριθμητικά την παραπάνω ποσότητα με την μέθοδο Markov Chain Monte Carlo (MCMC).

α) Υπολογίστε αναλυτικά το παραπάνω άθροισμα και επιβεβαιώστε ότι ο κώδικας προσφέρει μια καλή εκτίμησή του.

β) Αλλάξτε τον κώδικα ώστε να υπολογίζει το παραπάνω άθροισμα 50 φορές, και βρείτε την διασπορά των αποτελεσμάτων.

γ) Αν θέλαμε να περιορίσουμε την διασπορά των αποτελεσμάτων στο μισό, πόσο μεγάλο θα έπρεπε να πάρουμε το N; Απαντήστε θεωρητικά και επιβεβαιώστε το αριθμητικά.

δ) Αλλάξτε τον κώδικα ώστε να υπολογίζει με την μέθοδο MCMC το άθροισμα

$$\sum_{k=0}^{\infty} \frac{\cos(k + \cos(k))}{2^k}.$$

Άσκηση 2 Στην άσκηση αυτή θα χρησιμοποιήσουμε τον αλγόριθμο Metropolis-Hastings για να κάνουμε δειγματοληψία από την αναλλοίωτη κατανομή π του μοντέλου Ising. Καταβάστε και ανοίξτε τον κώδικα `ising.py`. Ο κώδικας αυτός προσομοιώνει ένα σύστημα από σωματίδια τοποθετημένα σε ένα πλέγμα $L \times L$, καθένα από τα οποία έχει σπιν +1 ή -1. Η ενέργεια μιας διαμόρφωσης από σπιν σ στο πλέγμα είναι

$$E(\sigma) = -\sum_{x \sim y} \sigma(x)\sigma(y).$$

Ας εξετάσουμε την συνεισφορά κάθε όρου του αθροίσματος. Κάθε όρος αντιστοιχεί σε γειτονικά σωματίδια x, y στο πλέγμα. Αν τα σωματίδια έχουν το ίδιο σπιν, τότε $\sigma(x)\sigma(y) = 1$ και ο όρος έχει αρνητική συνεισφορά στην ολική ενέργεια της διαμόρφωσης. Αντίθετα, αν τα σωματίδια έχουν αντίθετο σπιν, τότε $\sigma(x)\sigma(y) = -1$ και ο όρος αυξάνει την ολική ενέργεια της διαμόρφωσης. Σε θερμοδυναμική ισορροπία στην θερμοκρασία T, η πιθανότητα να βρούμε το σύστημα στην διαμόρφωση σ είναι

$$\pi(\sigma) = \frac{1}{Z(\beta)} e^{-\beta E(\sigma)}$$

όπου $\beta = 1/kT$ και η ποσότητα $Z(\beta) = \sum_\sigma e^{-\beta E(\sigma)}$ είναι ένας παράγοντας κανονικοποίησης που κάνει την π κατονομή. Παρατηρήστε ότι όσο πιο χαμηλή είναι η ενέργεια μιας διαμόρφωσης, τόσο πιο πιθανό είναι να βρούμε το σύστημα σε αυτήν την διαμόρφωση, ενώ υπάρχουν δύο διαμορφώσεις που ελαχιστοποιούν την ενέργεια: όλα τα σπιν +1, και όλα τα σπιν -1.

Παρατηρήστε επίσης ότι ο χώρος όλων των δυνατών διαμορφώσεων είναι ο $\mathbb{X} = \{-1, +1\}^{L \times L}$, αφού καθένα από τα $L \times L$ σωματίδια μπορεί να έχει σπιν +1 ή -1. Στον κώδικά μας θεωρούμε $L = 64$ οπότε ο χώρος των δυνατών διαμορφώσεων έχει πληθάριθμο $2^{4096} > 10^{1200}$, οπότε καταλαβαίνει κανείς ότι η δειγματοληψία από έναν τόσο μεγάλο χώρο είναι πρακτικά αδύνατη με συμβατικές μεθόδους. Ο κώδικας χρησιμοποιεί τον αλγόριθμο Metropolis-Hastings για να προσομοιώσει μια μαρκοβιανή αλυσίδα στον $\mathbb{X}$ που ξεκινά από μια τυχαία αρχική διαμόρφωση, και έχει κατανομή ισορροπίας την π.

α) Τρέξτε τον κώδικα για την δοσμένη θερμοκρασία $T = 30$ και δείτε την διαμόρφωση στην οποία καταλήγει ο αλγόριθμος.

Επαναλάβετε μερικές φορές.

β) Αλλάξτε τον κώδικα ώστε η αρχική διαμόρφωση των σπιν να είναι όλα $+1$. Ξανατρέξτε τον κώδικα μερικές φορές. Αλλάζουν τα ποιοτικά χαρακτηριστικά της διαμόρφωσης που επιστρέφει ο αλγόριθμος;

γ) Αρχίστε τώρα να κατεβάζετε την θερμοκρασία, θέτοντας διαδοχικά $T = 10.0, 3.0, 1.0, 0.3, 0.1, 0.03$ και περιγράψτε τι παρατηρείτε.

δ) Εξηγήστε γιατί η γραμμή του κώδικα

```
if random.uniform(0.0,1.0) < math.exp(-deltaE/Temperature):  Spin[k] *=-1
```

υλοποιεί τον αλγόριθμο Metropolis-Hastings.

δ) Για την θερμοκρασία $T = 0.03$ αλλάξτε την τιμή της παραμέτρου `nsteps`. Πώς μοιάζουν τα ενδιάμεσα στάδια από τα οποία περνάει η αλυσίδα για μέχρι να καταλήξουμε στην εικόνα που πήραμε για `nsteps` $= 100 \times L \times L$;

Άσκηση 3 Στην άσκηση αυτή θα χρησιμοποιήσετε την μέθοδο της προσομοιωμένης ανόπτησης για να βρείτε το ελάχιστο μιας συνάρτησης $V : \mathbb{R} \to \mathbb{R}$. Ο αλγόριθμος έχει ως εξής. Αρχικά το σωματίδιό μας τοποθετείται τυχαία στο διάστημα (-3.5,+3.5) σε θερμοκρασία $T = 128$. Σε κάθε βήμα επιχειρεί μια μετατόπιση που ακολουθεί ομοιόμορφη κατανομή στο διάστημα $(-\delta, +\delta)$ και πραγματοποιεί την μετατόπιση σύμφωνα με τον αλγόριθμο Metropolis-Hastings. Κάθε 100 βήματα ρίχνουμε την θερμοκρασία και επαναλαμβάνουμε μέχρι η θερμοκρασία να πέσει κάτω από το $T = 0.01$.

Κατεβάστε τον κώδικα `sim_annealing.py` και δείτε τον. Η παράμετρος `minima` σας επιτρέπει να επιλέξετε αν θα ελαχιστοποιήσετε μια συνάρτηση με 1, 2 ή 3 τοπικά ακρότατα. Η αρχική επιλογή είναι 3. Η παράμετρος `gamma` σας επιτρέπει να ελέγξετε τον ρυθμό ψύξης. Η αρχική επιλογή είναι να ρίχνουμε την θερμοκρασία στο μισό κάθε 100 βήματα. Η παράμετρος `iterations` σας επιτρέπει να επαναλάβετε την διαδικασία εύρεσης του ελαχίστου όσες φορές θέλετε. Η αρχική επιλογή της είναι 1. Η παράμετρος `delta` ελέγχει το μέγεθος της επιχειρούμενης μετατόπισης σε κάθε βήμα. Η αρχική της επιλογή είναι 3. Τέλος η λογική παράμετρος `animation` ελέγχει αν θέλουμε να δούμε μια οπτική αναπαράσταση της προσομοιωμένης ανόπτησης. Η αρχική της επιλογή είναι `True`.

α) Τρέξτε τον κώδικα. Παρατηρήστε πώς σε μεγάλες θερμοκρασίες το σωματίδιο κινείται σχετικά ελεύθερα στο χώρο, και πώς καθώς ψύχουμε το σύστημα το σωματίδιο εντοπίζεται κοντά στο ολικό ελάχιστο της V. Αλλάξτε την παράμετρο `minima` και επαναλάβετε.
β) Αλλάξτε την παράμετρο `iterations` σε 1000 και την παράμετρο `animation` σε `False` ώστε να επιταχύνετε το τρέξιμο του προγράμματος. Για τις διάφορες τιμές της παραμέτρου `minima` βρείτε σε τι ποσοστό επιτυγχάνει ο αλγόριθμος να βρει το ελάχιστο της V.
γ) Αλλάξτε την τιμή της παραμέτρου `delta` σε 0.1 και επαναλάβετε το προηγούμενο ερώτημα. Ποιο είναι τώρα το ποσοστό επιτυχούς εύρεσης του ολικού ελαχίστου; Γιατί πιστεύετε ότι συμβαίνει αυτό; Αν θέλετε να έχετε μια εικόνα του τι συμβαίνει, αλλάξτε την παράμετρο `iterations` σε 1 και την παράμετρο `animation` σε `True` για να δείτε πώς κινείται το σωματίδιο, και τρέξτε τον κώδικα μέχρι να δείτε το σωματίδιο να καταλήγει κοντά σε ένα τοπικό ελάχιστο, διαφορετικό από το ολικό.
δ) Αλλάξτε τον κώδικα ώστε να βρίσκει το ελάχιστο της συνάρτησης

$$V(x) = 10(x^2 - 1)^2 + 25(x - 1)^2.$$

Άσκηση 4 Στην άσκηση αυτή θα χρησιμοποιήσουμε τον αλγόριθμο της προσομοιωμένης ανόπτησης για να λύσουμε το πρόβλημα του πλανόδιου πωλητή. Κατεβάστε το αρχείο `eu.csv` που περιέχει το γεωγραφικό μήκος και πλάτος 27 ευρωπαϊκών πόλεων. Κατεβάστε τους κώδικες `simulated_annealing_tsp.py` και `plot_lib.py`. Ο πρώτος περιλαμβάνει την βασική ρουτίνα που θα χρησιμοποιήσουμε, ενώ ο δεύτερος εισάγεται σαν βιβλιοθήκη και χρησιμοποιείται για τα διαγράμματα που θα κατασκευάσουμε. Ο κώδικας `simulated_annealing_tsp.py` διαβάζει από το αρχείο `eu.csv` τις πόλεις και τα γεωγραφικά μήκη και πλάτη τους και χρησιμοποιεί την ρουτίνα `geodesic_distance` για να υπολογίσει την γεωδεσιακή τους απόσταση (την απόστασή τους πάνω στην γήινη σφαίρα). Στην συνέχεια χρησιμοποιεί τον αλγόριθμο της προσομοιωμένης ανόπτησης για να υπολογίσει μια προσεγγιστική λύση για το πρόβλημα του πλανόδιου πωλητή, δηλαδή με ποια σειρά πρέπει να επισκεφτεί κανείς αυτές τις πόλεις ώστε να ελαχιστοποιήσει την συνολική απόσταση που θα διανύσει.

Στην έξοδό του ο κώδικας επιστρέφει 3 διαγράμματα. Το πρώτο έχει ένα μονοπάτι όπου οι επισκέψεις στις πόλεις γίνονται με τυχαία σειρά. Το δεύτερο έχει το βέλτιστο μονοπάτι που υπολογίζει ο αλγόριθμος της προσομοιωμένης ανόπτησης, ενώ το τρίτο δείχνει πως εξελίσσεται καθώς κατεβάζουμε την θερμοκρασία η συνολική απόσταση για το τρέχον μονοπάτι του αλγορίθμου (μπλε γραμμή), και για το βέλτιστο μονοπάτι που έχει βρεθεί μέχρι τότε (κόκκινη γραμμή).

α) Τρέξτε τον κώδικα `simulated_annealing_tsp.py` και δείτε το μονοπάτι που επιστρέφει ο αλγόριθμος της προσομοιωμένης ανόπτησης, και την εξέλιξη της συνολικής απόστασης καθώς ψύχουμε το σύστημα. Τυπώστε αυτά τα διαγράμματα

και εξηγήστε την εικόνα που βλέπετε στο δεύτερο.
β) Εντοπίστε στον κώδικα τις εντολές που υλοποιούν τον αλγόριθμο `Metropolis-Hastings` και εξηγήστε πώς ακριβώς αλλάζει το μονοπάτι σε κάθε βήμα του αλγορίθμου.
γ) Αλλάξτε τον κώδικα ώστε να διαβάζει από το αρχείο `europe.csv` τις συντεταγμένες 35 ευρωπαϊκών πόλεων και λύστε το πρόβλημα του πλανόδιου πωλητή που θέλει να επισκεφτεί και τις 35 αυτές πόλεις. Στο διάγραμμα δεν τυπώνονται τα ονόματα των πόλεων για να είναι πιο ευκρινές το μονοπάτι.
δ) Πόσα είναι τα δυνατά μονοπάτια που περνούν από όλες τις 35 πόλεις μια μόνο φορά; Αν σε έναν υπολογιστή ο χρόνος που απαιτείται για τον υπολογισμό του συνολικού μήκους ενός τέτοιου μονοπατιού ήταν 1μsec πόσο χρόνο θα χρειαζόμασταν για να βρούμε το βέλτιστο μονοπάτι με έναν εξαντλητικό αλγόριθμο που θα υπολόγιζε την συνολική απόσταση που θα διανύσει ο πλανόδιος πωλητής κατά μήκος κάθε μονοπατιού και θα επέστρεφε το μονοπάτι με την ελάχιστη συνολική απόσταση;