Καθηγητής Π. Λουρίδας

Τμήμα Διοικητικής Επιστήμης και Τεχνολογίας

Οικονομικό Πανεπιστήμιο Αθηνών

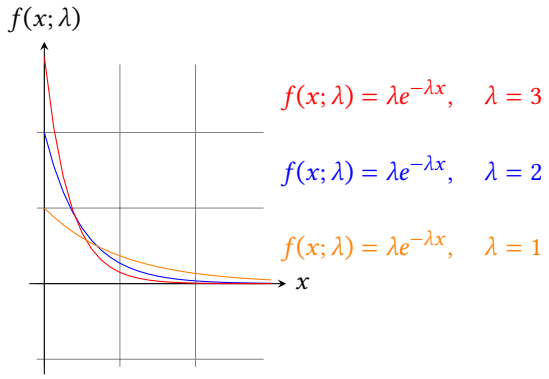
Εντοπισμός Εξάρσεων

Κάτι που μπορεί να έχετε παρατηρήσει και στην καθημερινή σας δραστηριότητα είναι ότι τα μηνύματα που αποστέλνονται στα διάφορα κοινωνικά δίκτυα, ή που λαμβάνετε στο ηλεκτρονικό σας ταχυδρομείο, δεν εμφανίζονται με τον ίδιο σταθερό ρυθμό. Συνήθως υπάρχουν περίοδοι ηρεμίας, οι οποίες διακόπτονται από περιόδους εξάρσεων, όπου καταιγιζόμαστε από τα τεκταινόμενα. Μάλιστα και οι ίδιες οι περίοδοι εξάρσεων δεν είναι ίδιες. Μπορεί να διανύουμε μια περίοδο ηρεμίας, η οποία να διακοπεί από μια περίοδο έξαρσης, η οποία με τη σειρά της να διακοπεί από μια περίοδο ακόμα μεγαλύτερης έξαρσης, κ.λπ.

Έστω τώρα ότι παρατηρούμε τα μηνύματα που αναρτώνται σε κάποιο κοινωνικό μέσο ή ομάδα, ή που παραδίδονται μέσω ηλεκτρονικού ταχυδρομείου. Αν έχουμε μια αποτύπωση της χρονικής στιγμής που αναρτάται ή φτάνει κάθε μήνυμα στον προορισμό του, έχουμε στα χέρια μας ένα ιστορικό των αλληλεπιδράσεων. Από το ιστορικό αυτό, μπορούμε να εντοπίσουμε τις περιόδους εξάρσεων, στη διάρκεια του χρόνου; Αν ναι, ένας ιστορικός του μέλλοντος, ή ένας ερευνητής, μπορεί να μελετήσει το σύστημα με βάση μια τέτοια χρονοσειρά γεγονότων. Θα μπορεί να βγάλει συμπεράσματα για τη συμπεριφορά του συστήματος, δηλαδή των χρηστών που έστελναν τα μηνύματα στις διάφορες χρονικές περιόδους.

Για να το κάνουμε αυτό, πρέπει να ξεκινήσουμε μοντελοποιώντας με κάποιον τρόπο το κομμάτι του κόσμου που μας ενδιαφέρει—αυτό κάνουμε πάντα στην επιστήμη. Θέλουμε λοιπόν ένα μοντέλο περιγραφής της εκπομπής μηνημάτων. Για την χρονική περιγραφή της εκπομπής των μηνυμάτων χρησιμοποιούμε την εκθετική κατανομή. Η εκθετική κατανομή περιγράφει διαδικασίες όπου εμφανίζονται γεγονότα συνεχώς, ανεξαρτήτως το ένα από το άλλο, με έναν σταθερό μέσο ρυθμό. Η συνάρτηση πυκνότητας πιθανότητας της εκθετικής κατανομής είναι:

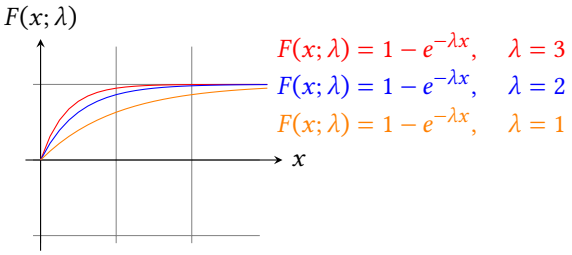
$$f(x; \lambda) = \lambda e^{-ax}$$



Η αναμενόμενη, ή μέση τιμή, της κατανομής είναι ίση με λ^{-1}, και αυτός είναι ο μέσος χρόνος μεταξύ των γεγονότων. Έτσι, η παράμετρος λ μας επιτρέπει να ρυθμίσουμε το πλάτος των διαστημάτων μεταξύ των γεγονότων: όσο μεγαλύτερο είναι το λ, τόσο μικρότερο είναι το διάστημα αυτό και τόσο μεγαλύτερη είναι η συχνότητα των γεγονότων.

Σύμφωνα με τον παραπάνω ορισμό, η πιθανότητα εμφάνισης ενός γεγονότος εντός χρονικού διαστήματος x δίνεται από την αθροιστική συνάρτηση κατανομής:

$$F(x;\lambda) = 1 - e^{-\lambda x}$$



Αντίστοιχα, η πιθανότητα το διάστημα που μεσολαβεί μέχρι την πραγματοποίηση ενός γεγονότος να υπερβεί το x είναι:

$$1 - F(x;\lambda) = e^{-\lambda x}$$

Στην πιο απλή περίπτωση, το σύστημα που παρακολουθούμε βρίσκεται σε μία κατάσταση q_0 όπου τα μηνύματα εμφανίζονται με την εκθετική κατανομή με παράμετρο λ_0:

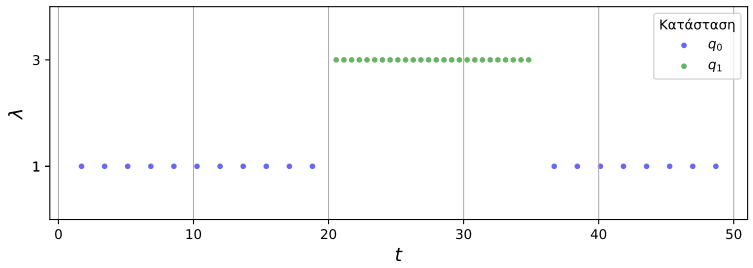
$$f_0(x) = \lambda_0 e^{-\lambda_0 x}$$

Μπορεί όμως τα μηνύματα να μην ακολουθούν πάντοτε την ίδια κατανομή μεταξύ των εμφανίσεων τους: σε κάποιες χρονικές περιόδους, τα μηνύματα μπορούν να εμφανίζονται με πιο γοργό ρυθμό. Αυτό μπορούμε να το περιγράψουμε ορίζοντας ότι το

σύστημα μπορεί να βρίσκεται σε μία από δύο διαφορετικές καταστάσεις. Η πρώτη κατάσταση είναι αυτή που περιγράψαμε, ενώ στη δεύτερη κατάσταση q_1 τα μηνύματα εκπέμπονται σύμφωνα με την εκθετική κατανομή με μία άλλη παράμετρο $\lambda_1 > \lambda_0$:

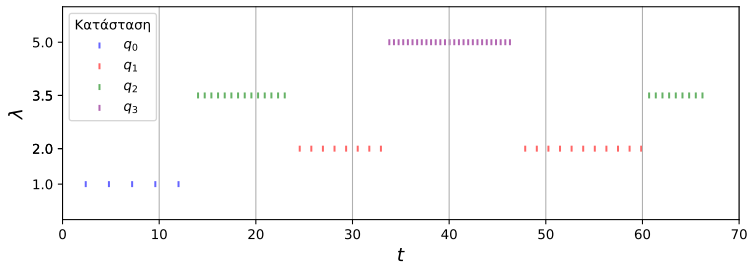
$$f_1(x) = \lambda_1 e^{-\lambda_1 x}$$

Αν πάρουμε λοιπόν τον άξονα του χρόνου, αυτό σημαίνει ότι σε κάποιες περιόδους, στις οποίες το σύστημα βρίσκεται στην κατάσταση q_1, θα παρατηρούμε πιο έντονη δραστηριότητα.



Μπορούμε να γενικεύσουμε το μοντέλο αυτό, ώστε το σύστημα να έχει οσεσδήποτε καταστάσεις, οι οποίες αντιστοιχούν σε διαφορετικούς ρυθμούς εκπομπής των μηνυμάτων. Τότε σε κάθε κατάσταση q_i, τα μηνύματα θα εκπέμπονται σύμφωνα με την εκθετική κατανομή:

$$f_i(x) = \lambda_i e^{-\lambda_i x}$$



Έστω ότι έχουμε $n + 1$ μηνύματα τα οποία εμφανίζονται μέσα σε ένα χρονικό διάστημα T, και τα μηνύματα αυτά εμφανίζονται με n χρονικά διαστήματα μεταξύ τους, $x_1, x_2, \ldots, x_n$: x_1 είναι το χρονικό διάστημα που μεσολαβεί μεταξύ των δύο πρώτων μηνυμάτων, x_2 είναι το χρονικό διάστημα που μεσολαβεί μεταξύ του δεύτερου και του τρίτου μηνύματος, κ.ο.κ. Τότε αποδεικνύεται ότι το σύστημα μπορεί να περιγραφεί με k καταστάσεις $\{q_0, q_1, \ldots, q_{k-1}\}$, όπου:

$$k = \lceil 1 + \log_s T + \log_s(1/\min(\{x_1, x_2, \ldots, x_n\})) \rceil$$

Η κάθε μία κατάσταση ορίζεται από την εκθετική κατανομή σύμφωνα με την οποία εκπέμπονται μηνύματα όσο το σύστημα βρίσκεται σε αυτήν. Αν $g = T/n$, η παράμετρος λ_i της εκθετικής κατανομής της κατάστασης q_i θα είναι:

$$\lambda_i = s^i/g$$

Στον παραπάνω ορισμό, $s > 1$ είναι μια παράμετρος που ορίζουμε εμείς για να προσαρμόζουμε τις εκθετικές κατανομές. Οι καταστάσεις $q_0, q_1, \dots q_{k-1}$ αντιστοιχούν σε διαστήματα μεταξύ εκπομπών με ρυθμό που αυξάνεται ακολουθώντας γεωμετρική πρόοδο με λόγο $1/s$. Πράγματι, ο μέσος χρόνος μεταξύ εκπομπών στην κατάσταση q_i είναι λ_i^{-1}, στην κατάσταση q_{i+1} είναι λ_{i+1}^{-1}, και $\lambda_{i+1}^{-1}/\lambda_i^{-1} = s^i/s^{i+1} = s^{-1}$.

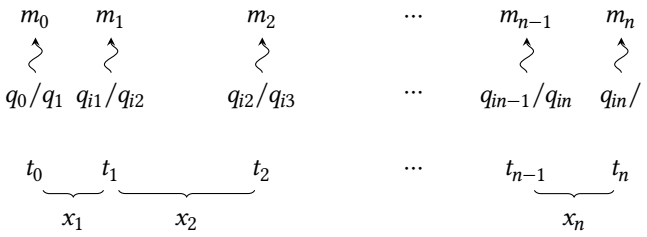
Το σύστημα μπορεί να μεταβαίνει από μια κατάσταση σε μία άλλη, δηλαδή από μια περίοδο δραστηριότητας σε μία άλλη. Η μετάβαση από μια περίοδο δραστηριότητας σε μια περίοδο πιο υψηλής δραστηριότητας απαιτεί ένα κόστος. Αντίθετα, η μετάβαση από μια περίοδο υψηλής δραστηριότητας σε μια περίοδο χαμηλότερης δραστηριότητας δεν απαιτεί κάποιο κόστος. Η παραμονή του συστήματος στην ίδια δραστηριότητα, επίσης δεν απαιτεί κάποιο κόστος. Επομένως το κόστος της μετάβασης σε μια κατάσταση (που μπορεί να είναι η ίδια) είναι:

$$\tau(i,j) = \begin{cases} \gamma(j-i)\ln n & \text{αν } j > i \\ 0 & \text{αν } j \leq i \end{cases}$$

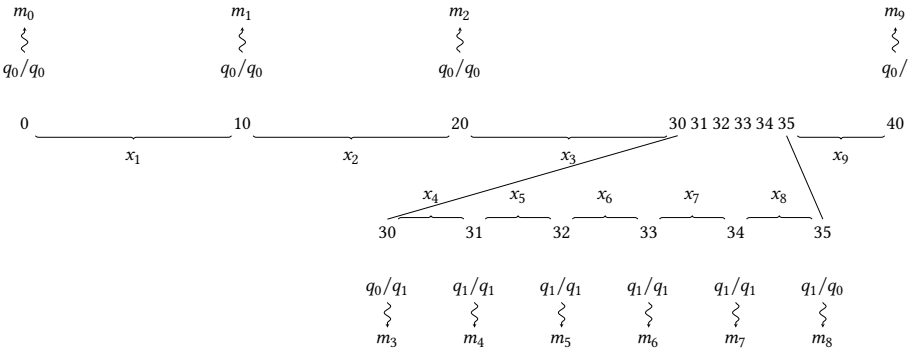
Στον παραπάνω ορισμό, $\gamma > 0$ είναι μια παράμετρος που ορίζουμε εμείς ώστε να ρυθμίζουμε το κόστος της μετάβασης από μια κατάσταση σε μια άλλη.

Το πρόβλημα που έχουμε εμείς είναι να εκμαιεύσουμε τη λειτουργία ενός συστήματος αν στη διάθεσή μας έχουμε μόνο τις εκπομπές των μηνυμάτων. Δηλαδή, αν έχουμε μια σειρά $n+1$ μηνυμάτων που εκπέπονται μέσα σε ένα συνολικό χρόνο T, μπορούμε να συνάγουμε σε τι κατάσταση βρισκόταν το σύστημα κατά την εκπομπή του κάθε μηνύματος; Αν ναι, τότε θα μπορέσουμε να εντοπίσουμε περιόδους εξάρσεων, οι οποίες θα είναι αυτές στις οποίες το σύστημα εκπέμπει μηνύματα με μεγαλύτερη συχνότητα, δηλαδή το σύστημα βρίσκεται σε υψηλή κατάσταση δραστηριότητας.

Για να λύσουμε το πρόβλημα, θα πρέπει να βρούμε την ακολουθία των καταστάσεων στις οποίες βρέθηκε το σύστημα στη διάρκεια T της εκπομπής των μηνυμάτων. Το σύστημα ξεκινάει τη στιγμή t_0 στην κατάσταση q_0, όπου εκπέμπει το πρώτο μήνυμα m_0 και αμέσως μεταβαίνει σε μία κατάσταση q_{i1}, $i1 \in [0, k-1]$. Μετά από χρόνο x_1, ο οποίος καθορίζεται από την εκθετική κατανομή στην κατάσταση q_{i1}, εκπέμπει το δεύτερο μήνυμα m_1 τη χρονική στιγμή t_1 και μεταβαίνει σε μία κατάσταση q_{i2}, $i2 \in [0, k-1]$. Πάλι, μετά από χρόνο x_2, ο οποίος καθορίζεται από την εκθετική κατανομή στην κατάσταση q_{i2} εκπέμπει το τρίτο μήνυμα m_2 τη χρονική στιγμή t_2. Συνεχίζει με τον τρόπο αυτό ώστε μετά από συνολικά n διαστήματα βρίσκεται σε μία κατάσταση q_{in} οπότε εκπέμπει το τελευταίο μήνυμα m_n τη χρονική στιγμή t_n.



Για παράδειγμα, έστω ότι ένα σύστημα εκπέμπει μηνύματα τις χρονικές στιγμές $[0, 10, 20, 30, 31, 32, 33, 34, 35, 40]$, διαγραμματικά θα ήταν όπως παρακάτω:



Αφού η μετάβαση από κατάσταση σε κατάσταση μπορεί να επιφέρει κόστος, ορίζουμε το $C_j(t)$ ως το ελάχιστο κόστος που μπορεί να έχει μια ακολουθία καταστάσεων για την εκπομπή των μηνυμάτων σε διαστήματα $x_1, x_2, \ldots, x_t$. Το $C_j(t)$ υπολογίζεται ως εξής:

- Αν $t = 0$, $C_0(0) = 0$ και $C_j(0) = \infty$ για $j > 0$.
- Αν $t > 0$, $C_j(t) = -\ln f_j(x_t) + \min_\ell(C_\ell(t-1) + \tau(\ell, j))$.

Η πρώτη συνθήκη σημαίνει ότι τη χρονική στιγμή 0 το σύστημα βρίσκεται στην κατάσταση q_0 με μηδενικό κόστος, ενώ το κόστος για κάθε άλλη κατάσταση είναι άπειρο.

Για να καταλάβουμε τη δεύτερη συνθήκη, ας πάρουμε κάθε έναν όρο του αθροίσματος με τη σειρά. Για τον πρώτο όρο, $-\ln f_j(x_t)$, πρέπει σκεφτούμε ότι τη χρονική στιγμή $t > 0$, το σύστημα βρίσκεται στην κατάσταση q_j, οπότε η πιθανότητα να εκπέμψει μήνυμα μετά από χρόνο x_t ακολουθεί την εκθετική κατανομή $f_j(x_t)$. Εμείς θέλουμε να επιλέξουμε την κατάσταση q_j που ταιριάζει όσο το δυνατόν περισσότερο με το διάστημα x_t που περνάει για να εκπέμψει το μήνυμα m_j. Αυτό μας το δίνει ο πρώτος όρος.

Για τον δεύτερο όρο, προκειμένου να βρίσκεται στην κατάσταση j, το σύστημα θα έχει συνολικά κόστος ίσο με το κόστος που είχε στην προηγούμενη κατάσταση q_ℓ, συν το κόστος μετάβασης από την ℓ στην j. Και αφού θέλουμε το ελάχιστο συνολικό

κόστος, θέλουμε το άθροισμα $C_\ell(t-1)+\tau(\ell,j)$ να είναι το ελάχιστο δυνατό, με άλλα λόγια, η q_ℓ θα είναι αυτή που ελαχιστοποιεί το άθροισμα.

Ο υπολογισμός του κόστους με αυτόν τον τρόπο γίνεται με μια εφαρμογή ενός διαδεδομένου αλγορίθμου, του αλγορίθμου Viterbi, την οποία μπορείτε να δείτε παρακάτω.

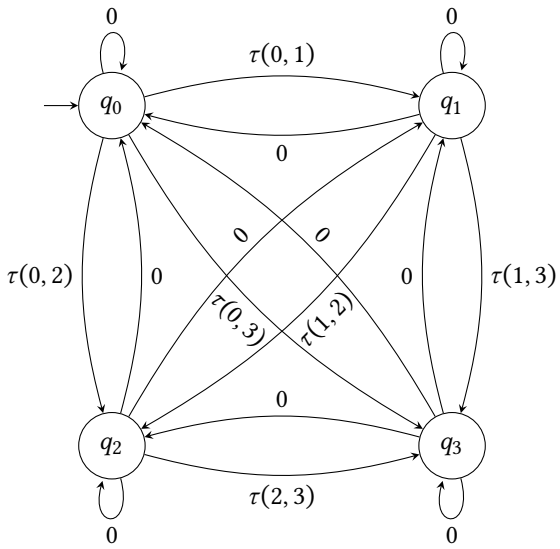
`BurstsViterbi(`X`)`$\to P$

Input: X, ένας πίνακας μεγέθους n που περιέχει τα διαστήματα μεταξύ διαδοχικών εκπομπών
Data: C, ένας πίνακας $(n+1)\times k$ με $C[0,0]=0$ και $C[i,j]=\infty$ για $i,j\neq 0$
P, ένας πίνακας $k\times(n+1)$ με $P[i,j]=0$
Output: S, ένας πίνακας μεγέθους $n+1$ που περιέχει την ακολουθία των καταστάσεων του συστήματος

```
1  for t = 1 to n + 1 do
2      for s = 0 to k do
3          ℓ_min ← 0
4          c_min ← C[t − 1, 0] + τ(0, s)
5          for ℓ = 1 to k do
6              c ← C[t − 1, ℓ] + τ(ℓ, s)
7              if c < c_min then
8                  c_min ← c
9                  ℓ_min ← ℓ
10         C[t, s] ← c_min − ln f_s(X[t − 1])
11         P[s, 0:t] ← P[ℓ_min, 0:t]
12         P[s, t] ← s
13 c_min ← C[n, 0]
14 s_min ← 0
15 for s = 1 to k do
16     if C[n, s] < c_min then
17         c_min ← C[n, s]
18         s_min ← s
19 return P[s_min]
```

Ένα σύστημα όπως αυτό που περιγράψαμε, το οποίο μπορεί να μεταβαίνει από κατάσταση σε κατάσταση αντιστοιχεί σε μία *μηχανή πεπερασμένων καταστάσεων* (finite state machine) ή *αυτόματο πεπερασμένων καταστάσεων* (finite state automaton). Ένα τέτοιο αυτόματο αποτελείται από ένα πεπερασμένο σύνολο καταστάσεων και τις δυνατές μεταβάσεις μεταξύ τους. Η λειτουργία του ξεκινάει από μια από αυτές τις καταστάσεις, η οποία είναι η αρχική κατάσταση, και μπορεί να έχει και ένα σύνολο τελικών καταστάσεων. Οι καταστάσεις, οι μεταβάσεις, και το κόστος της μετάβασης από τη μία κατάσταση στην άλλη μπορούν να περιγραφούν με ένα γράφο, όπως ο παρακάτω γράφος περιγράφει ένα σύστημα τεσσάρων καταστάσεων. Όταν το σύστημα βρίσκεται σε μία κατάσταση, εκπέμπει ένα μήνυμα μετά από χρόνο που δίνεται

από την εκθετική συνάρτηση $f_i(x)$ και μεταβαίνει σε μία άλλη κατάσταση (που μπορεί να είναι η ίδια με αυτήν που βρίσκεται) με κόστος ίσο με το βάρος του συνδέσμου του γράφου. Ένα σύστημα όπως αυτό που περιγράψαμε που εκπέμπει μηνύματα σε τέσσερεις διαφορετικές καταστάσεις, ξεκινώντας από την κατάσταση q_0, και χωρίς τελική κατάσταση, αναπαρίσταται από το παρακάτω πεπερασμένο αυτόματο:



Αν τώρα θέλουμε να αναπαραστήσουμε την εξέλιξη του συστήματος στο χρόνο, μπορούμε να χρησιμοποιήσουμε έναν διαφορετικό γράφο. Ο γράφος αυτός θα έχει ένα κόμβο για κάθε κατάσταση σε κάθε χρονική στιγμή, δηλαδή το σύνολο των κόμβων του θα είναι:

$$V = \{(t_c, q_i),\ c \in \{0, 1, \dots, n\},\ i \in \{0, 1, \dots k-1\}\}$$

Κάθε κόμβος από t_1 και μετά συνδέεται με όλους τους κόμβους την επόμενη χρονική στιγμή, με το βάρος που προκύπτει από το αντίστοιχο κόστος, εκτός από τη χρονική στιγμή t_0 όπου μόνο ο κόμβος (q_0, t_0) συνδέεται με τους κόμβους τη χρονική στιγμή t_1 (γιατί το σύστημα ξεκινάει τη χρονική στιγμή t_0). Άρα, το σύνολο των συνδέσμων του γράφου θα είναι το

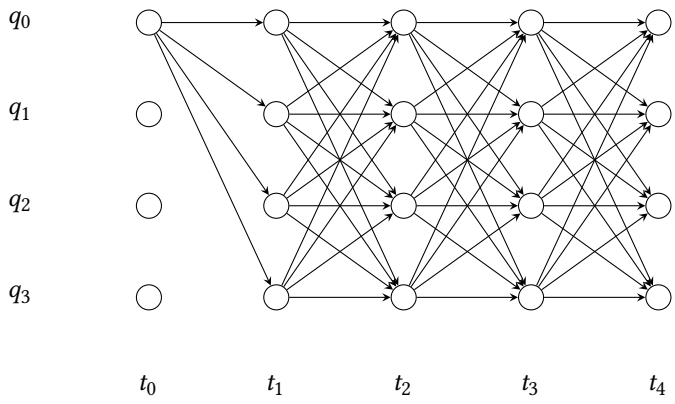
$$E = \{((t_0, q_0), (t_1, q_j)),\ j \in \{0, 1, \dots, k-1\}\}$$

$$\cup$$

$$\{((t_c, q_i), (t_{c+1}, q_j)),\ c \in \{1, \dots, n-1\},\ i, j \in \{0, 1, \dots, k-1\}\}$$

Αφού ο κάθε σύνδεσμος αναπαριστά τη μετάβαση από μία κατάσταση q_i σε μία κατάσταση q_j σε δύο διαδοχικές χρονικές στιγμές t_c και t_{c+1}, θα έχει βάρος ίσο το κόστος

της μετάβασης $\tau(i, j)$ συν το κόστος για την εκπομπή μηνύματος $-\ln f_j(t_{c+1} - t_c)$. Παρακάτω βλέπετε έναν τέτοιο γράφο για τέσσερες καταστάσεις και πέντε χρονικές στιγμές.



Ο γράφος αυτός ονομάζεται *καφασωτό* (trellis). Τότε, για να βρούμε τη βέλτιστη ακολουθία καταστάσεων του συστήματος αρκεί να βρούμε τα συντομότερο μονοπάτι από τον κόμβο (t_0, q_0) στους κόμβους (t_n, q_j). Αυτό μπορεί να γίνει με οποιονδήποτε αλγόριθμο εύρεσης συντομότερων μονοπατιών, αλλά λόγω της μορφής του γράφου βολεύει ο αλγόριθμος Bellman-Ford.

Σκοπός της εργασίας είναι η εύρεση της ακολουθίας καταστάσεων του συστήματος και των περιόδων που το σύστημα βρίσκεται σε κάθε επίπεδο δραστηριότητας, χρησιμοποιώντας τους δύο διαφορετικούς τρόπους που περιγράφηκαν, δηλαδή τον αλγόριθμο Viterbi και τον αλγόριθμο Bellman-Ford.

Απαιτήσεις Προγράμματος

Κάθε φοιτητής θα εργαστεί σε αποθετήριο στο GitHub. Για να αξιολογηθεί μια εργασία θα πρέπει να πληροί τις παρακάτω προϋποθέσεις:

- Για την υποβολή της εργασίας θα χρησιμοποιηθεί το ιδιωτικό αποθετήριο του φοιτητή που δημιουργήθηκε για τις ανάγκες του μαθήματος και του έχει αποδοθεί. Το αποθετήριο αυτό έχει όνομα του τύπου `username-algo-assignments`, όπου `username` είναι το όνομα του φοιτητή στο GitHub. Για παράδειγμα, το σχετικό αποθετήριο του διδάσκοντα θα ονομαζόταν `louridas-algo-assignments` και θα ήταν προσβάσιμο στο https://github.com/dmst-algorithms-course/louridas-algo-assignments. *Τυχόν άλλα αποθετήρια απλώς θα αγνοηθούν.*
- Μέσα στο αποθετήριο αυτό θα πρέπει να δημιουργηθεί ένας κατάλογος `assignment-2024-2`.
- Μέσα στον παραπάνω κατάλογο το πρόγραμμα θα πρέπει να αποθηκευτεί με το όνομα `bursts.py`.

- Δεν επιτρέπεται η χρήση έτοιμων βιβλιοθηκών γράφων ή τυχόν έτοιμων υλοποιήσεων των αλγορίθμων, ή τμημάτων αυτών, εκτός αν αναφέρεται ρητά ότι επιτρέπεται.
- Επιτρέπεται η χρήση δομών δεδομένων της Python όπως στοίβες, λεξικά, σύνολα, κ.λπ.
- Επιτρέπεται η χρήση των παρακάτω βιβλιοθηκών ή τμημάτων τους όπως ορίζεται:
  - `sys.argv`
  - `argparse`
  - `math`
  - `collections.deque`
- Το πρόγραμμα θα πρέπει να είναι γραμμένο σε Python 3.
- Η εργασία είναι αποκλειστικά ατομική. Δεν επιτρέπεται συνεργασία μεταξύ φοιτητών στην εκπόνησή της, με ποινή το μηδενισμό. Επιπλέον η εργασία δεν μπορεί να είναι αποτέλεσμα συστημάτων Τεχνητής Νοημοσύνης (όπως ChatGPT). Ειδικότερα όσον αφορά το τελευταίο σημείο προσέξτε ότι τα συστήματα αυτά χωλαίνουν στην αλγοριθμική σχέση, άρα τυχόν προτάσεις που κάνουν σε σχετικά θέματα μπορεί να είναι λανθασμένες. Επιπλέον, αν θέλετε να χρησιμοποιήσετε τέτοια συστήματα, τότε αν και κάποιος άλλος το κάνει αυτό, μπορεί οι προτάσεις που θα λάβετε να είναι παρόμοιες, οπότε οι εργασίες θα παρουσιάσουν ομοιότητες και άρα θα μηδενιστούν.
- Η έξοδος του προγράμματος θα πρέπει να περιλαμβάνει μόνο ό,τι φαίνεται στα παραδείγματα που παρατίθενται. *Η φλυαρία δεν επιβραβεύεται.*

Χρήση του GitHub

Όπως αναφέρθηκε το πρόγραμμά σας για να αξιολογηθεί θα πρέπει να αποθηκευθεί στο GitHub. Επιπλέον, θα πρέπει το GitHub να χρησιμοποιηθεί καθόλη τη διάρκεια της ανάπτυξης του.

Αυτό σημαίνει ότι *δεν θα ανεβάσετε στο GitHub απλώς την τελική λύση του προβλήματος* μέσω της λειτουργίας "Upload files". Στο GitHub θα πρέπει να φαίνεται *το ιστορικό της συγγραφής του προγράμματος.* Αυτό συνάδει και με τη φιλοσοφία του εργαλείου: λέμε "commit early, commit often". Εργαζόμαστε σε ένα πρόγραμμα και κάθε μέρα, ή όποια στιγμή έχουμε κάνει κάποιο σημαντικό βήμα, αποθηκεύουμε την αλλαγή στο GitHub. Αυτό έχει σειρά ευεργετικών αποτελεσμάτων:

- Έχουμε πάντα ένα αξιόπιστο εφεδρικό μέσο στο οποίο μπορούμε να ανατρέξουμε αν κάτι πάει στραβά στον υπολογιστή μας (μας γλιτώνει από πανικούς του τύπου: ένα βράδυ πριν από την παράδοση ο υπολογιστής μας ή ο δίσκος του πνέει τα λοίσθια, και εμείς τι θα υποβάλουμε στον Λουρίδα;).
- Καθώς έχουμε πλήρες ιστορικό των σημαντικών αλλαγών και εκδόσεων του προγράμματός μας, μπορούμε να επιστρέψουμε σε μία προηγούμενη αν συνει-

δητοποιήσουμε κάποια στιγμή ότι πήραμε λάθος δρόμο (μας γλιτώνει από πανικούς του τύπου: μα το πρωί δούλευε σωστά, τι στο καλό έκανα και τώρα δεν παίζει τίποτε;).

- Καθώς φαίνεται η πρόοδος μας στο GitHub, επιβεβαιώνουμε ότι το πρόγραμμα δεν είναι αποτέλεσμα της όποιας επιφοίτησης (μας γλιτώνει από πανικούς του τύπου: καλά, πώς το έλυσες το πρόβλημα αυτό με τη μία, μόνος σου;).
- Αν δουλεύετε σε μία ομάδα *που βεβαίως δεν είναι καθόλου η περίπτωση μας εδώ*, μπορούν όλοι να εργάζονται στα ίδια αρχεία στο GitHub εξασφαλίζοντας ότι ο ένας δεν γράφει πάνω στις αλλαγές του άλλου. Παρά το ότι η εργασία αυτή είναι ατομική, καλό είναι να αποκτάτε τριβή με το εργαλείο git και την υπηρεσία GitHub μιας και χρησιμοποιούνται ευρέως και όχι μόνο στη συγγραφή κώδικα.

Άρα επενδύστε λίγο χρόνο στην εκμάθηση των git / GitHub, των οποίων ο σωστός τρόπος χρήσης είναι μέσω γραμμής εντολών (command line), ή ειδικών προγραμμάτων (clients) ή μέσω ενοποίησης στο περιβάλλον ανάπτυξης (editor, IDE).

Τελικό Πρόγραμμα

Το πρόγραμμα θα καλείται ως εξής (όπου `python` η κατάλληλη εντολή στο εκάστοτε σύστημα):

```
python bursts.py [-s S] [-g GAMMA] [-d] {viterbi,trellis} offsets_file
```

Η σημασία των παραμέτρων του προγράμματος είναι:

- `-s S` αν δίνεται, η τιμή της παραμέτρου s του αλγορίθμου. Αν δεν δίνεται, θεωρούμε $s = 2$.
- `-s GAMMA` αν δίνεται, η τιμή της παραμέτρου γ του αλγορίθμου. Αν δεν δίνεται, θεωρούμε $\gamma = 1$.
- `-d` αν δίνεται το πρόγραμμα θα εκτυπώνει στην έξοδο τα διαγνωστικά μηνύματα που θα δείτε στα παραδείγματα που ακολουθούν.
- `{viterbi,trellis}` αν ο χρήστης δίνει `viterbi`, το πρόγραμμα θα εκτελεί τον αλγόριθμο Viterbi· αν ο χρήστης δίνει `trellis`, το πρόγραμμα θα εκτελεί τον αλγόριθμο Bellman-Ford.
- `offsets_file` είναι το όνομα του αρχείου που περιέχει τις χρονικές στιγμές εκπομπής των μηνυμάτων.

Σκοπός της εργασίας είναι η συγγραφή προγράμματος που θα εντοπίζει τις περιόδους δραστηριότητας ενός συστήματος με τον αλγόριθμο που περιγράψαμε. *Υλοποιήσεις άλλων αλγορίθμων δεν είναι αποδεκτές.*

Παραδείγματα

Παράδειγμα 1

Αν ο χρήστης του προγράμματος δώσει:

```
python bursts.py viterbi two_states.txt
```

το πρόγραμμά σας θα διαβάσει το αρχείο two_states.txt. Το αρχείο αυτό περιέχει τις χρονικές στιγμές στις οποίες το σύστημα εξέπεμψε μηνύματα. Στην έξοδο το πρόγραμμά σας θα πρέπει να εμφανίσει ακριβώς τα παρακάτω:

```
0 [0.0 30.0)
1 [30.0 35.0)
0 [35.0 40.0)
```

Αυτό σημαίνει ότι το σύστημα βρίσκεται στην κατάσταση 0 από τη χρονική στιγμή 0 μέχρι τη χρονική στιγμή 30, τη χρονική στιγμή 30 μεταπίπτει στην κατάσταση 1, και τη χρονική στιγμή 35 επιστρέφει στην κατάσαση 0. Επομένως, εντοπίσαμε μία περίοδο αυξημένης δραστηριότητας στο διάστημα από τη χρονική στιγμή 30 έως τη χρονική στιγμή 35.

Παράδειγμα 2

Αν ο χρήστης του προγράμματος δώσει:

```
python bursts.py viterbi two_states.txt -d
```

Όπως και πριν, το πρόγραμμά σας θα διαβάσει το αρχείο two_states.txt. Στην έξοδο το πρόγραμμά σας θα πρέπει να εμφανίσει ακριβώς τα παρακάτω (στρογγυλοποιούμε στα δύο δεκαδικά ψηφία):

```
[0, inf, inf, inf, inf, inf, inf]
[3.74, 7.5, 13.5, 24.0, 43.51, 81.01, 154.52]
[7.48, 11.24, 17.24, 27.75, 47.25, 84.75, 158.26]
[11.22, 14.98, 20.98, 31.49, 50.99, 88.5, 162.0]
[12.94, 14.67, 16.62, 19.03, 22.33, 27.44, 36.14]
[14.66, 15.92, 17.63, 20.03, 23.34, 28.44, 37.15]
[16.37, 17.17, 18.64, 21.04, 24.34, 29.45, 38.15]
[18.09, 18.42, 19.64, 22.04, 25.35, 30.45, 39.16]
[19.81, 19.66, 20.65, 23.05, 26.35, 31.46, 40.16]
[22.28, 22.71, 25.25, 31.26, 41.76, 61.26, 98.77]
10 [0, 0, 0, 0, 1, 1, 1, 1, 1, 0]
0 [0.0 30.0)
1 [30.0 35.0)
0 [35.0 40.0)
```

Οι πρώτες δέκα γραμμές δίνουν, με ακρίβεια δύο δεκαδικών ψηφίων, την αρχική τιμή της γραμμής μηδέν του πίνακα C και τη γραμμή t του πίνακα C στο τέλος κάθε επανάληψης των γραμμών 1–12 του αλγορίθμου.

Στη συνέχεια εμφανίζεται μια γραμμή με τον αριθμό και τις καταστάσεις στις οποίες βρίσκεται σε κάθε χρονική στιγμή το σύστημα, και ακολουθεί η υπόλοιπη έξοδος του προγράμματος όπως προηγουμένως.

Παράδειγμα 3

Αν ο χρήστης του προγράμματος δώσει:

```
python bursts.py trellis two_states.txt
```

το πρόγραμμά σας θα διαβάσει το αρχείο το ίδιο αρχείο όπως και πριν και θα εμφανίσει την ίδια έξοδο, πλην όμως χρησιμοποιώντας τον αλγόριθμο Bellman-Ford:

```
0 [0.0 30.0)
1 [30.0 35.0)
0 [35.0 40.0)
```

Παράδειγμα 4

Αν ο χρήστης του προγράμματος δώσει:

```
python bursts.py trellis two_states.txt -d
```

το πρόγραμμά σας θα λειτουργήσει όπως στο προηγούμενο παράδειγμα, αλλά θα εμφανίσει επιπλέον στην έξοδο την εξέλιξη των χαλαρώσεων (relaxations) κατά την εκτέλεση του αλγορίθμου Bellman Ford. Έτσι, η γραμμή:

```
(9, 5) 62.48 -> 61.26 from (8, 2) 20.65 + 6.59 + 34.03
```

δείχνει ότι το κόστος του μονοπατιού για τον κόμβο `(9, 5)`, που αντιστοιχεί στην κατάσταση q_5 τη χρονική στιγμή t_9, χαλάρωσε από `62.48` σε `62.21` προερχόμενοι από τον κόμβο `(8, 2)`, που αντιστοιχεί στην κατάσταση q_2 την προηγούμενη χρονική στιγμή t_8. Το μονοπάτι μέχρι τον `(8, 2)` έχει κόστος `20.65`, ενώ `6.59` είναι το κόστος της μετάβασης μεταξύ των κόμβων και `34.03` το κόστος εκπομπής. Οι αριθμοί θα εμφανίζονται με ακρίβεια δύο δεκαδικών ψηφίων. Για λόγους οικονομίας χώρου παρακάτω εμφανίζονται μόνο οι δύο πρώτες και οι δύο τελευταίες γραμμές των χαλαρώσεων—το πρόγραμμά σας θα τις εμφανίζει όλες.

Στη συνέχεια εμφανίζεται μια γραμμή με τον αριθμό και τις καταστάσεις στις οποίες βρίσκεται σε κάθε χρονική στιγμή το σύστημα, και ακολουθεί η υπόλοιπη έξοδος του προγράμματος όπως προηγουμένως.

```
(1, 0) inf -> 3.74 from (0, 0) 0.00 + 0.00 + 3.74
(1, 1) inf -> 7.50 from (0, 0) 0.00 + 2.20 + 5.30
...
(9, 5) 62.48 -> 61.26 from (8, 2) 20.65 + 6.59 + 34.03
(9, 6) 99.98 -> 98.77 from (8, 2) 20.65 + 8.79 + 69.33
10 [0, 0, 0, 0, 1, 1, 1, 1, 1, 0]
0 [0.0 30.0)
1 [30.0 35.0)
0 [35.0 40.0)
```

Παράδειγμα 5

Αν ο χρήστης του προγράμματος δώσει:

```
python bursts.py viterbi three_states_1.txt
```

το πρόγραμμά σας θα διαβάσει το αρχείο three_states_1.txt και θα εμφανίσει στην έξοδο ακριβώς τα παρακάτω:

```
0 [0.0 410.0)
1 [410.0 450.0)
2 [450.0 469.0)
1 [469.0 600.0)
0 [600.0 710.0)
1 [710.0 800.0)
0 [800.0 1000.0)
```

Παράδειγμα 6

Αν ο χρήστης του προγράμματος δώσει:

```
python bursts.py trellis three_states_1.txt
```

το πρόγραμμά σας θα διαβάσει πάλι το αρχείο three_states_1.txt και θα εμφανίσει στην έξοδο τα ίδια όπως στο προηγούμενο παράδειγμα, χρησιμοποιώντας τον αλγόριθμο Bellman-Ford:

```
0 [0.0 410.0)
1 [410.0 450.0)
2 [450.0 469.0)
1 [469.0 600.0)
0 [600.0 710.0)
1 [710.0 800.0)
0 [800.0 1000.0)
```

Παράδειγμα 7

Αν ο χρήστης του προγράμματος δώσει:

```
python bursts.py viterbi three_states_2.txt -s 3 -g 0.5
```

το πρόγραμμά σας θα διαβάσει το αρχείο three_states_2.txt και θα εμφανίσει στην έξοδο τα παρακάτω:

```
0 [0.0 400.0)
1 [400.0 450.0)
2 [450.0 470.0)
1 [470.0 601.0)
0 [601.0 700.0)
1 [700.0 800.0)
0 [800.0 1000.0)
```

Παράδειγμα 8

Αν ο χρήστης του προγράμματος δώσει:

```
python bursts.py trellis three_states_2.txt -s 3 -g 0.5
```

το πρόγραμμά σας θα διαβάσει το αρχείο του προηγούμενου παραδείγματος και θα βγάλει την ίδια έξοδο, χρησιμοποιώντας τον αλγόριθμο Bellman-Ford:

```
0 [0.0 400.0)
1 [400.0 450.0)
2 [450.0 470.0)
1 [470.0 601.0)
0 [601.0 700.0)
1 [700.0 800.0)
0 [800.0 1000.0)
```

Παράδειγμα 9

Αν ο χρήστης του προγράμματος δώσει:

```
python bursts.py viterbi four_states.txt -s 1.1 -g 0.025
```

το πρόγραμμά σας θα διαβάσει το αρχείο four_states.txt και θα εμφανίσει στην έξοδο τα παρακάτω:

```
0 [2.4 14.02)
2 [14.02 22.99)
0 [22.99 33.81)
6 [33.81 46.29)
0 [46.29 59.87)
1 [59.87 66.21)
```

Προσέξτε ότι, αφού οι καταστάσεις του συστήματος αντιστοιχούν σε συγκεκριμένες εκθετικές κατανομές, δεν είναι απαραίτητο οι καταστάσεις που προκύπτουν να έχουν μεταξύ τους διαφορά ένα.

Παράδειγμα 9

Αν ο χρήστης του προγράμματος δώσει:

```
python bursts.py trellis four_states.txt -s 1.1 -g 0.025
```

το πρόγραμμά σας θα διαβάσει το αρχείο του προηγούμενου παραδείγματος και θα εμφανίσει την ίδια έξοδο, χρησιμοποιώντας όμως τον αλγόριθμο Bellman-Ford:

```
0 [2.4 14.02)
2 [14.02 22.99)
0 [22.99 33.81)
6 [33.81 46.29)
0 [46.29 59.87)
1 [59.87 66.21)
```

Περισσότερες Πληροφορίες

Η έρευνα για τα πρότυπα αλληλεπίδρασης στα κοινωνικά μέσα, και η οποία για τον εντοπισμό των εξάρσεων χρησιμοποιεί την προσέγγιση Viterbi που παρουσιάσαμε εδώ, είναι η ς [1]. Η προσέγγιση αυτή για την εντοπισμό εξάρσεων προτάθηκε από τον Jon Kleinberg [2], ο οποίος τον δοκίμασε στην προσωπική ηλεκτρονική του αλληλογραφία. Ο αλγόριθμος Viterbi πήρε το όνομά του από τον Andrew Viterbi που τον παρουσίασε το 1967 [3], αν και έχει εφευρεθεί ανεξάρτητα από διάφορους ερευνητές· στοιχεία για τον αλγόριθμο μπορείτε να δείτε στη σχετική σελίδα της Wikipedia.

Βιβλιογραφία

[1] Michele Avalle et al. "Persistent interaction patterns across social media platforms and over time". In: *Nature* 628.8008 (2024), pp. 582–589. DOI: 0.1038/s41586-024-07229-y.

[2] Jon Kleinberg. "Bursty and Hierarchical Structure in Streams". In: *Data Mining and Knowledge Discovery* 7.4 (2003), pp. 373–397. DOI: 10.1023/A:1024940629314.

[3] Andrew J. Viterbi. "Error Bounds for Convolutional Codes and an Asymptotically Optimum Decoding Algorithm". In: *IEEE Transactions on Information Theory* 13.2 (1967), pp. 260–269. DOI: 10.1109/TIT.1967.1054010.

Καλή Επιτυχία!