Aleksa Tešić

Predikcija sekvencijalnog kretanja u 2D ravni

Predložena su i analizirana dva algoritma za predikciju kretanja agenta zasnovana na skrivenom Markovljevom modelu (SMM). Posmatrana su dva slučaja kretanja: u slobodnoj 2D ravni i 2D lavirintu. Kod kretanja u lavirintu ispitivana je predikcija za prvi naredni korak, a kod kretanja u ravni za čitavu sekvencu. Tačnost predikcije za lavirint iznosi 65%, dok je pri predikciji čitave sekvence kretanja uočeno da model tačno predviđa samo sekvence kretanja sa konstantim promenama stanja. Upoređena je i vremenska efikasnost jednonitne i paralelizovane implementacije algoritma. Pokazano je da je pri radu sa velikim matricama paralelizovana implementacija algoritma u svakom segmentu brža od jednonitne implementacije. Razlike u brzinama su se, u zavisnosti od operacija koje su bile izvršavane, kretale od 1.6 do 8 puta bržeg izvršavanja u korist paralelizovane implementacije.

Uvod

Potreba za predikcijom događaja koji se dešavaju kako u diskretnim, tako i kontinualnim vremenskim intervalima, je čest problem na koji se nailazi u računarstvu. Jedan od načina za rešavanje ovakvih problema je korišćenje skrivenog Markovljevog modela (u daljem tekstu SMM). SMM je tehnika mašinskog učenja koja je, između ostalog, namenjena predikciji sekvencijalnog kretanja (Bishop 2006), a primenu nalazi i u bioinformatici (Ivanović 2016), prepoznavanju rukopisa (Brown *et al*. 1996), predviđanju kretanja na berzi (Hassan i Nath 2005), predviđanju kretanja pešaka (Asahara *et al*. 2011) itd.

Cilj ovog rada je ispitivanje tačnosti predikcije kretanja agenta koji se nalazi u 2D prostoru, u dva slučaja: kretanja u ravni bez prepreka i kretanja u lavirintu. Za kretanje u ravni agent je konstantno ponvaljao istu sekvencu kretanja, a zatim je ova sekvenca korišćena i za treniranje modela. Nakon toga pokušano je rekreiranje čitave sekvence slobodnog kretanja u ravni. Za kretanje u lavirintu model je obučavan na 400 sekvenci kretanja koje je agent generisao, a svaka sekvenca sadržala je 400 koraka. Nakon toga sledilo je predviđanje sledećeg najverovatnijeg koraka agenta. Određen je najbolji odnos između broja iteracija potrebnih za treniranje modela i tačnosti predikcije.

U oba slučaja upoređene su vremenske efikasnosti dve različite implementacije metoda. Jedna implementacija je bila pomoću programa koji koristi paralelno izvršavanje, a druga pomoću programa koji ga ne koristi.

Metode

SMM je matematički model namenjen predviđanju događaja koji se izvršavaju u diskretnom vremenu. U ovom radu SMM je predviđao sekvencijalno kretanje nekog posmatranog agenta. SMM kao ulaz dobija niz stanja kroz koje je posmatrani agent prošao. Taj niz stanja ćemo obeležiti sa X. Na osnovu dobijenih n stanja SMM predviđa sledeće stanje, odnosno stanje sa indeksom $n+1$.

Svakom članu niza X pridružena je skrivena promenljiva iz niza Z, koja može biti u jednom od unapred definisanih stanja.

Pomoću ova dva niza kreirane su transmisiona matrica T, emisiona matrica E i vektor P.

Aleksa Tešić (1999), Žiča 134a, učenik 4. razreda Gimnazije u Kraljevu

MENTOR: Miloš Savić, Prirodno-matematički fakultet Univerziteta u Novom Sadu

Transmisiona matrica predstavlja verovatnoću da skrivena promenljiva iz niza Z sa indeksom $n+1$ pređe u stanje j ako je promenljiva sa indeksom n u stanju i. Dimenzije matrice T su $a \times a$ gde je a ukupan broj skrivenih stanja koje može imati jedna promenljiva niza Z.

Emisiona matrica predstavlja verovatnoću koju skrivena promenljive iz niza X ima da bude u nekom stanju, pod uslovom da je stanje promenljive z iz Z, koja joj odgovara, poznato. Dimenzije E matrice su $b \times a$ gde je b broj mogućih stanja koje može imati jedan član niza X, dok je a ukupan broj skrivenih stanja koje može imati jedan član niza Z.

Vektor *P* predstavlja raspodelu inicijalnih verovatnoća da element niza Z uzme neko od unapred definisanih k stanja.

Matrice se u procesu inicijalizacije postavljaju na nasumične normalizovane vrednosti, dok se elementi niza P svi postavljaju na jednake, takođe normalizovane vrednosti. Zatim se, primenom EM (Expectation Maximization) algoritma (Bishop 2006) određuju konačne vrednosti matrice E koje su nam potrebne za predviđanje.

Algoritam maksimizacije (Expectation-Maximization)

EM algoritam (Bishop 2006) se koristi za fitovanje parametara modela na osnovu datog skupa za obuku. Pod parametrima se podrazumevaju skrivene promenljive iz niza Z, na osnovu kojih se zatim menjaju E, T i P. Kao ulaz, pored skupa za obuku, EM dobija dva parametra:

- broj mogućih stanja koje može imati skrivena promenljiva iz Z
- broj iteracija u kojima se fituju parametri modela

Potrebno je naći najbolji odnos između ova dva parametra, tako da ne dolazi do overfitovanja. Do overfitovanja dolazi kada posle određenog broja stanja za Z i određenog broja iteracija, tačnost predviđanja krene da opada. To se može primetiti tek nakon završetka predikcije (slika 1).

Predviđanje čitave sekvence kretanja

Za rekreiranje izgleda sekvence od početka do kraja korišćen je Viterbi algoritam (Bishop 2006). Napravljen je poseban pomoćni program koji je kao ulaz dobijao unapred izgenerisanu sekvencu u kojoj se ponavljao određen šablon kretanja. Dobijena sekvenca je zatim rekreirana korišćenjem SMM-a. Početna i predviđena sekvenca su zatim predstavljene grafički korišćenjem biblioteke Cairo (www.cairographics.org). Ukupno je generisano je 4 ovakve sekvence, a na osnovu njih su generisane 4 nove predviđene sekvence (slika 2).

Slično metodu za predviđanje prvog sledećeg koraka, broj skrivenih stanja za Z, broj posmatranih stanja za X i broj iteracija za trening su unapred poznati. Oni se mogu menjati da bi se utvrdilo za koje vrednosti se dobijaju najpreciziji rezultati.

Predviđanje prvog sledećeg koraka

U sklopu ovog rada je realizovan program za predikciju kretanja agenta unutar poznatog lavirinta (Jovanović 2015). Program je napisan u programskom jeziku C++ u razvojnom okruženju CodeBlocks. Takođe, napravljen je poseban pomoćni program (u daljem tekstu program za generisanje podataka za obuku) koji je uz korišćenje unapred određene strategije upravljao agentom. Strategija koju je agent koristio prilikom ovog istraživanja je da uvek pravi korak u desno, a ako ne može, onda nasumično bira jedno od preostalih mogućih koraka. Program za generisanje podataka za obuku je u tekstualnu datoteku ispisivao kretanja agenta, tako što je svako kretanje imalo svoje stanje:

0. stanje – kretanje ka gore
1. stanje – kretanje ka dole
2. stanje – kretanje na desno
3. stanje – kretanje na levo
4. stanje – nema kretanja

Generisano je 400 ovakvih datoteka, svaka sa 400 zabeleženih stanja. Program za predikciju kretanja je onda za svaku datoteku uzimao prvih 300 stanja koje je koristio kao skup za obuku, da bi zatim na osnovu treninga odredio 301. stanje. Kako je to 301. stanje unapred bilo poznato, tačnost predikcije je određivana poređenjem 301. stanja generisanog programom za obuku i 301. stanja koje je program za predikciju predvideo.

Paralelizacija

U ovom delu istraživanja, vršeno je ispitivanje vremenske efikasnosti programa pri radu sa matricama koje se zahtevaju i prilikom predikcije prvog sledećeg kretanja i prilikom predikcije čitave sekvence kretanja (odnosno i za kretanje kroz lavirint i kretanje u slobodnoj ravni). Ispitivana je efikasnost jednonitne implementacije metoda napisane upotrebom C++ standardne biblioteke i implementacije metoda koja koristi biblioteku Armadillo koja podržava paralelizaciju i koja već ima implementirane operacije sa matricama i vektorima.

Merenja su vršena uz pomoć funkcije clock() u okruženju Codeblocks. Operativni sistem na kom su merenja vršena je Linux Debian 9, dok je processor: Intel Core i5 3317U @ 1.70GHz. Uz ovo, kompajler je bio podešen na O2 opciju za ubrzavanje.

Rezultati

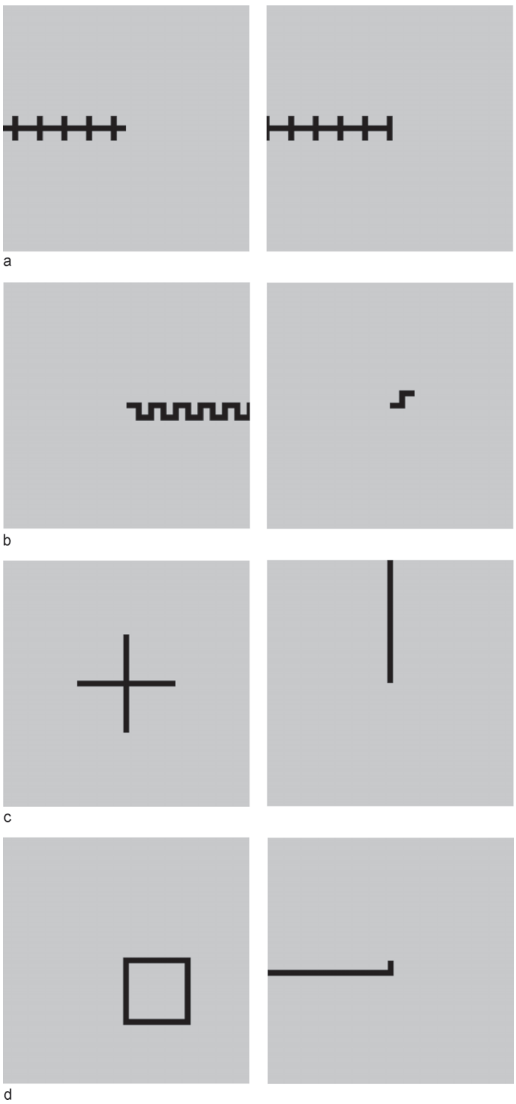
Slobodno kretanje u ravni: predviđanje čitave sekvence

Kako bi ovaj metod trebalo da predvidi čitavo sekvencijalno kretanje, napravljen je poseban program koji je na 2D prostoru iscrtavao sekvencijalno kretanje, koje je zatim uz pomoć Cairo biblioteke i grafički predstavljeno (slika 1).

Posmatranjem ovih primera, a na osnovu činjenice da je poznat proces iscrtavanja (što je u datom slučaju skup stanja od 0 do 3, gde svako stanje označava određen smer), dolazimo do zaključka da program uspešno kopira samo sekvence koje konstantno menjaju stanja (slika 1a), dok je kopiranje ostalih sekvenci neuspešno (slika 1: b, c i d).

Kretanje u lavirintu: predviđanje prvog narednog koraka

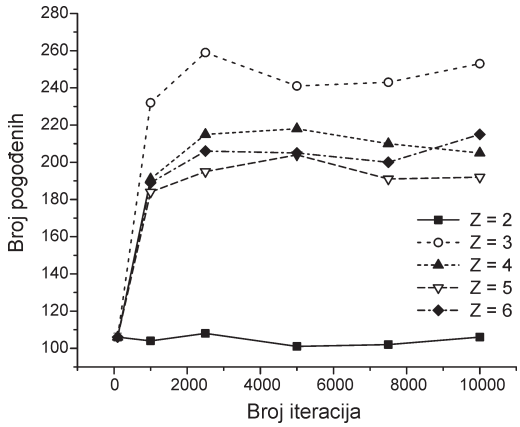
Na slici 1 je prikazana zavisnost broja uspešnih predikcija od broja iteracija za fitovanje SMM modela za različite vrednosti Z (broj stanja u kojima mogu biti skrivene promenljive). Iz dobijenih rezultata (slika 2) se može videti da za više od 3 stanja za Z, gde je Z jedna skrivena promenljiva, dolazi do overfitovanja. To nam pokazuje da nema smisla povećavati broj stanja za Z, jer će se preciznost samo gubiti, a vreme izvršavanja će svaki put biti sve veće.



Slika 1. Ilustracije uspešnosti predikcije: sa leve strane se nalaze očekivane sekvence kretanja koje je metod trebalo da predvidi; sa desne strane se nalaze sekvence kretanja koje je model zapravo predvideo.

Figure 1. Examples of expected and predicted sequences: on the left side are expected movement sequences that the model should have predicted; on the right are movement sequences actually predicted by the model.

Takođe, na istoj slici se primećuje da program najbolje rezultate daje na približno 2500



Slika 2. Uspešnost predikcije za različite vrednosti Z: broj pogođenih kretanja u zavisnosti od broja iteracija

Figure 2. Success rate of prediction for different Z values: number of successfully predicted movements depending on the number of iterations

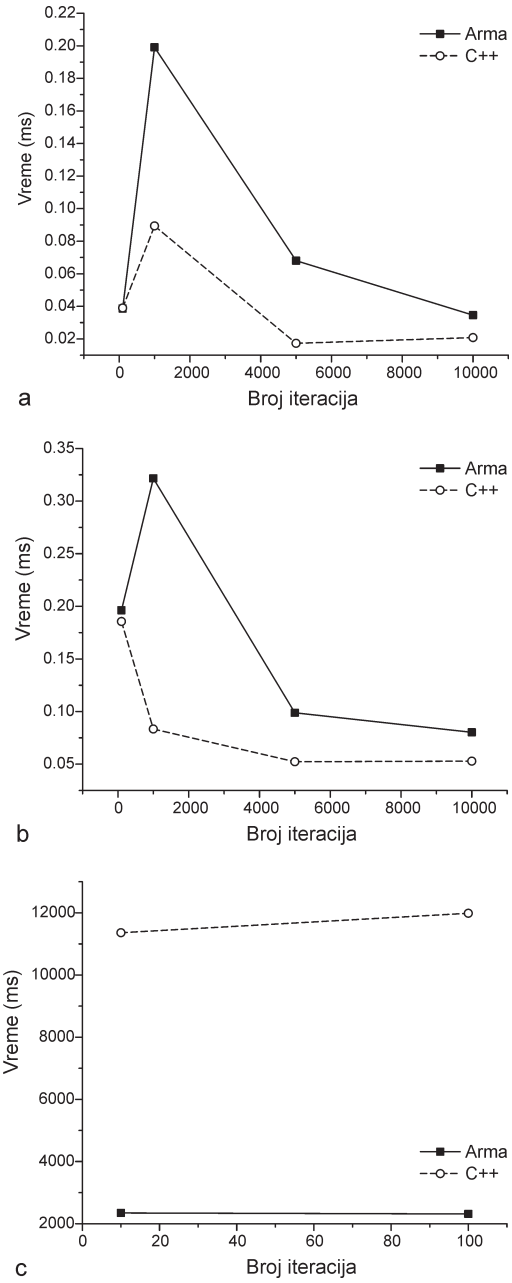
iteracija, i da nema potrebe povećavati broj iteracija na više, jer će se nauštrb još jednog ili dva pogođena odgovora izgubiti na brzini i dobiti veća vremenska i memorijska složenost. Iz grafika se može zaključiti da program u zavisnosti od broja stanja *Z* i broja iteracija ima uspešnost predviđanja između 25% i 65%.

Vremenska efikasnost

Kako je već poznato koje operacije sa matricama programi za predikciju koriste, merenja nisu vršena na celom kodu, nego samo na posebno napisanim funkcijama koje sadrže te operacije. Merenja su obuhvatala onaj deo koda koji se odnosi na te operacije, a zarad preciznosti, isti program je izvršavan više puta, a zatim se uzimalo prosečno vreme izvršavanja programa.

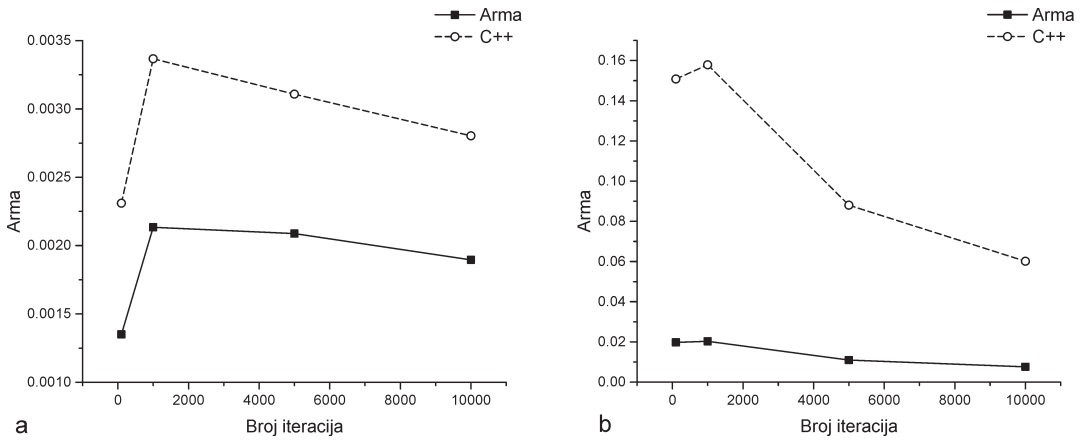
Ponavljanje merenja brzine izvršavanja i njihovo usrednjavanje je rađeno zato što funkcija clock() može vratiti različita vremena za izvršavanje istog dela koda. Taj efekat je uzrokovan drugim procesima koji se istovremeno sa ovim programom izvršavaju na procesoru. Merenja su izvršavana nad sledećim funkcijama:

1) Množenje matrica
2) Inicijalizacija matrica
3) Skalarni proizvod



Slika 3. Efikasnost množenja matrica: vreme izvršavanja operacije u zavisnosti od broja iteracija. Dimenzije matrica: a) 2 ×3, b) 5 ×5, c) 100 ×100.

Figure 3. Efficiency of matrix multiplications: time needed to complete the operation depending on the number of iterations. Matrix dimensions: a) 2 ×3, b) 5 ×5, c) 100 ×100.

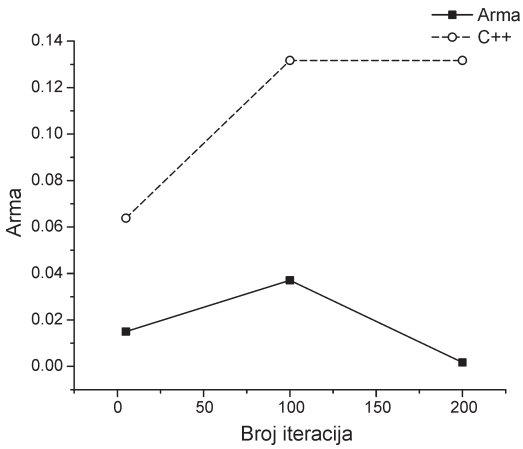


Slika 4. Efikasnost Inicijalizacije: vreme izvršavanja operacije u zavisnosti od broja iteracija, a) 5 ×5 matrice i b) 100 × 100 matrice.

Figure 4. Efficiency of matrix initialization: time needed to complete the operations depending on the number of iterations, a) 5 ×5 matrix and b) 100 × 100 matrix.

Množenje matrica. Merene su brzine prilikom množenja 2×3, 5×5 i 100×100 matrica. Sa datih grafika (slika 3) može se videti da je Armadillo brži pri radu sa velikim matricama.

Incijalizacija matrice. Merena je brzina inicijalizacije 5×5 i 100×100 matrice. Sa prikazanih grafika se može videti da je Armadillo biblioteka brža (slika 4).



Slika 5. Efikasnost računanja skalarnog proizoda: vreme izvršavanja operacije u zavisnosti od veličine matrice

Figure 5. Efficiency of dot product calculation: time needed to complete the operation depending on the size of the matrix

Skalarni proizvod. Merena je brzina izvršavanja ove operacije na matricama različitih veličina. Kao i u prethodnim primerima, i ovde se može videti da je Armadillo biblioteka brža od jednonitne implemetacije (slika 5).

Sa grafika se može uočiti da je pri radu sa velikim matricama implementacija programa uz pomoć Armadillo biblioteke od 1.6 do 8 puta brža od implementacije programa koji koristi standardnu C++ biblioteku.

Zaključak

U ovom radu je merena efikasnost predikcije sekvencijalnog kretanja u 2D ravni korišćenjem SMM-a. Ispitivana je tačnost predikcije prvog sledećeg koraka pri kretanju, kao i tačnost predikcije čitave sekvence kretanja. Takođe, ispitivana je vremenska efikasnost dve različite implementacije metoda, korišćenjem jednonitnog i paralelnog izvršavanja.

Predikcija cele sekvence kretanja se pokazala kao uspešna samo u slučajevima kojima se stanja unutar sekvence konstantno menjaju, dok je u ostalim slučajevima bila neuspešna. Predviđanje sekvencijalnog kretanja u lavirintu korišćenjem SMM-a se pokazalo kao uspešno. Uspešnost SMM-a pri predikcijama je bila u intervalu od 25% do 65%, gde je maksimum postignut za skrivene promenljive *Z* sa 3 moguća stanja, u

slučaju kada je korišćeno 2500 iteracija za fitovanje parametara.

Rezultati merenja vremenske efikasnosti su pokazali da se uz korišćenje biblioteke Armadillo mogu unaprediti brzine izvršavanja operacija, i da će ponovna implementacija programa za predikciju kretanja uz pomoć nje defintivno ubrzati računanje. Vremenska efikasnost programa koji je koristio Armadillo biblioteku je, u zavisnosti od toga koje su operacije nad matricama bile izvršavane, bila između 1.6 i 8 puta brža od standardne jednonitne implementacije.

Dalja istraživanja mogu biti usmerena na otkrivanje zašto metod ne predviđa dobro određene tipove sekvenci. Takođe, moguće je unaprediti evaluaciju predloženog metoda merenjem Menhetn distance između stvarne sekvence i sekvence kretanja koju dobijamo predikcijom. Moguće je i prilagoditi ceo program izvršavanju na GPU, a zatim uporediti vremenske efikasnosti takve implementacije sa trenutnom.

Literatura

Asahara A., Maruyama K., Sato A., Seto K. 2011. Pedestrian-movement prediction based on mixed Markov-chain model. U *Proceedings of the 19th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems, November 01-04, 2011*. New York: ACM, str. 25–33.

Anderson C., Curtin R. 2016. Armadillo: a template based c++ library for linear algebra. *Journal Of Open Source Software*, **1**: 26.

Bishop C. 2006. *Pattern Recognition and Machine Learning*. New York: Springer

Brown M. K., Turin W., Hu J. 1996. HMM based online handwriting recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **18**: 10.

Hassan M. R., Nath B. 2005. Stock market forecasting using hidden Markov model: a new approach. U *Proceedings, 5th International Conference on Intelligent Systems Design and Applications (ISDA'05)*. Washington: IEEE Computer Society, str. 192–196.

Ivanović M. 2016. Primena mašinskog učenja i skrivenih Markovljevih modela na prepoznavanje topologije transmembranskih proteina. Nepublikovan maturski rad. Matematička Gimnazija, Kraljice Natalije 37, 11000 Beograd.

Komogortsev O., Khan J. 2008. Eye movement prediction by Kalman filter with integrated linear horizontal oculomotor plant mechanical model. U *Proceedings of the 2008 symposium on Eye tracking research & applications*. New York: ACM, str. 229–236.

Jovanović N. 2015. Upoređivanje efikasnosti algoritama za snalaženje u lavirintu. *Petničke sveske*, 74: 205.

www.cairographics.org. Cairo Library: https://www.cairographics.org/

Code::Blocks: https://www.codeblocks.org/

Linux Debian 9: https://www.debian.org/News/2017/20170617

Clock() funkcija: http://www.cplusplus.com/reference/ctime/clock/

Intel Core i5 3317U @ 1.70GHz: https://ark.intel.com/products/65707/Intel-Core-i5-3317U-Processor-3M-Cache-up-to-2_60-GHz

Aleksa Tešić

Prediction of Sequential Movements in 2D Spaces

The goal of this project was testing two algorithms based on the Hidden Markov Model. The project consisted of predictions of the next most probable move of an agent in a 2D maze, and prediction of a whole sequence of movements in a 2D plane. For the prediction of the next possible state, the biggest achieved accuracy was 64.75%. Prediction of the whole sequence of movements was successful only with constantly changing states. Also, the time efficiencies of the singlethreaded and multithreaded implementation of the method were measured and compared. Time efficiency of the multithreaded implementation of the method proved to be better than the singlethreaded implementation when working with large matrices. Depending on the operation, the multithreaded program was between 1.6 and 8 times faster than the singlethreaded program.