PRIMENJENI ALGORITMI PSI

Januar 2022.

ISPITNA PITANJA

1. Evolutivni algoritmi

- Evolutivni algoritmi su algoritmi koji su bazirani na principima prirodne evolucije i nasleđivanja. U svakom koraku kreira se populacija jedinki. Svaka jedinka predstavlja potencijalno rešenje problema i implementirana je pomoću neke strukture podataka. Za svaku jedinku se izračunava prilagođenost (fitness) i selektuju se najpogodnije jedinke za sledeću generaciju, a one loše umiru ili bivaju eliminisane. Na neke jedinke mogu biti primenjeni genetski operatori ukrštanja i mutacije.

2. Genetski algoritam: Osnovni pojmovi i koraci

- **Genetski algoritam** predstavlja evolutivni algoritam zasnovan na principima evolucije.

  Osnovni pojmovi su:
  - **jedinka** ili **hromozom** koja je potencijalno rešenje problema
  - **gen** predstavlja deo jedinke koji se odnosi na deo podatka
  - **populacija** koja je skup više jedinki
  - **generacija** koja je skup svih jedinki u jednom trenutku
  - **prilagođenost** je vrednost kriterijuma optimalnosti
  - **operatori** su akcije nad jedinkama (ukštanje, mutacija, selekcija, elitizam)

  Koraci u genetskom algoritmu su:
  - **inicijalizacija početne populacije** najčešće nasumično
  - **izračunavanje prilagođenosti** za svaku jedinku u populaciji
  - **primena genetskih operatora** dok se uslovi ne zadovolje
  - **izbor jedinke sa najvećom fitness funkcijom**

3. Genetski algoritam: Kodiranje i tipovi kodiranja

- **Kodiranje** predstavlja transformaciju domenskog znanja u okvire genetskog algoritma čime utičemo na samu uspešnost algoritma. Kodiranje se projektuje zajedno sa operatorima.

  Tipovi kodiranja su:
  - **binarno kodiranje (**binarni sistem za predstavu podataka)
  - **realno kodiranje** (realni brojevi za predstavu podataka)
  - **uređeno kodiranje**

Binarno kodiranje predstavlja kodiranje jedinke tako što se formira N-bitni binarni string npr. 0011 0111 1111.
Ovakav tip kodiranja koristi se kod celobrojnih vrednosti kada nam tačnost nije presudni faktor te nam pojednostavljuje operacije ukrštanja i mutacije.

Takođe, problem koji se rešava primenom binarnog kodiranja je **problem ranca** gde je potrebno uzeti što više predmeta da se ne prekorači kapacitet ranca.

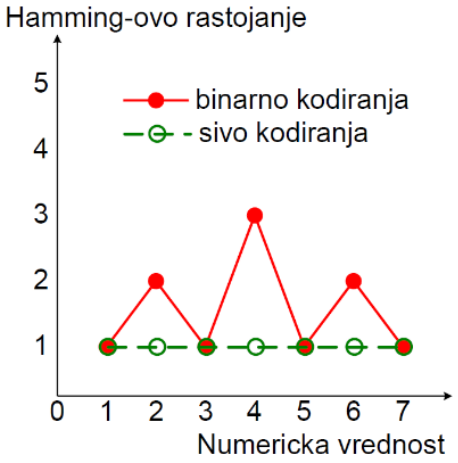
Binarno kodiranje: Problem Hamming-ove litice

Problem Hamingove litice se javlja kada su dve numerički bliske dok imaju veoma udaljene kodne oznake.

Kako bi se došlo do optimalnog rešenja potrebno je promeniti sve bite te ovaj problem prevazilazimo **Grejovim kodiranjem**.

Grejovo kodiranje

Primenom sivog kodiranja problem Hamingovog rastojanja između numerički susednih podataka poprima unificiranu vrednost **1**.



Koordianata	0	1	2	3	4	5	6	7
Bin. kod.	000	001	010	011	100	101	110	111
Sivo kod.	000	001	**011**	**010**	**110**	**111**	**101**	**100**

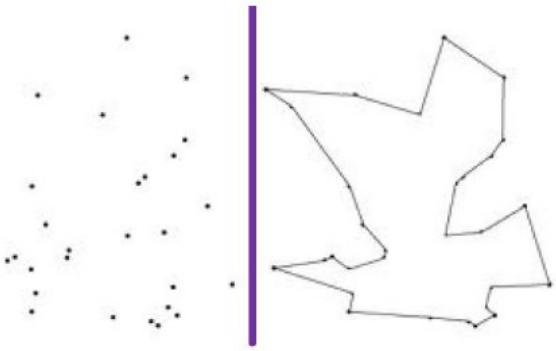
Realno kodiranje predstavlja kodiranje jedinke tako što se jedinka predstavlja realnim brojevima kada su binarne jedinke predugačke ali sa sobom povlači znatno složenije operacije ukrštanja i mutacije.

Redosledno kodiranje

Primena ovakvog tipa kodiranja doprinosi pri rešavanju **problema trgovačkog putnika**.

Problem trgovačkog putnika

Funkcija cilja je pronaći putanju posete uz minimalne troškove.



Ograničenja su da svi gradovi moraju biti posećeni samo jednom (sem početnog).

Euklidova udaljenost se može uzeti kao mera troškova.

Potrebno je pronaći najbolju putanju od **n!** mogućih putenja.

4. Genetski algoritam: Početna populacija

- **Početna populacija** se formira na osnovu zadate **veličine populacije** i **načina generisanja**. Za svaku od jedinki neophodno je izračunati i vrednost prilagođenosti tj. vrednost fitness funkcije.

Kolika će biti veličina populacije određuje se na osnovu toga da li nam je potrebno:

- više genetskog materijala, samim tim i veće šanse za dolaženje do optimalnog rešenja ali uz dobijanje sporijeg algoritma => **veća populacija**
- manje genetskog materijala uz prednost veće brzine algoritma => **manja populacija**
- **promenljiva populacija**

Načini generisanja početne populacije su:

- **slučajno generisanje** (najčešće se koristi)
- **uniformno (veštačko) generisanje** kojim se pokriva prostor pretrage

U početnu populaciju treba uključiti rešenja dobijena drugim optimizacionim algoritmima, ako su dostupna.

Fitness funkcija predstavlja kriterijum optimalnosti tj. izračunavanje stepena prilagođenosti za svaku jedinku.

Ako kriterijumska funkcija nema samo pozitivne vrednosti onda se vrši skaliranje/translacija.

5. Genetski algoritam: Karakteristike i vrste selekcije

- **Selekcija** određuje jedinke koje će ostati u sledećoj generaciji i obično se definiše tako da one jedinke, koje imaju bolji fitness, imaju veće šanse da budu izabrane.
  Ovo nije uvek najbolji izbor jer često može da uzrokuje **zaglavljivanje** u lokalnom optimumu.

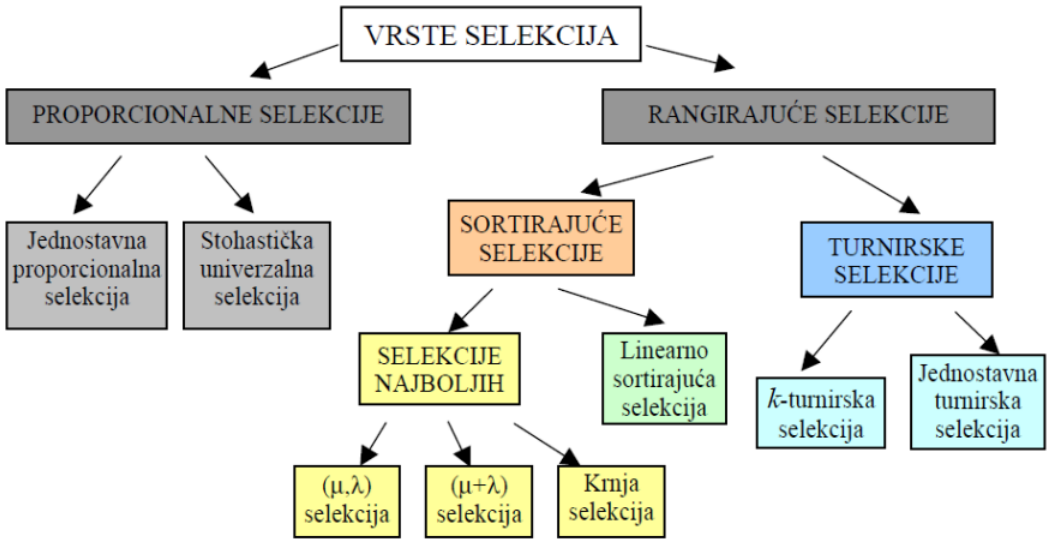
  Jedinke se mogu prenositi u sledeću generaciju na jedan od dva načina:
  - **generativno**
  - **eliminaciono**

  Svaka jedinka (sem najboljih) postoji samo u jednoj generaciji.

  Karakteristike selekcije mogu biti:
  - **brzina**
  - **pritisak**
  - **raznolikost**

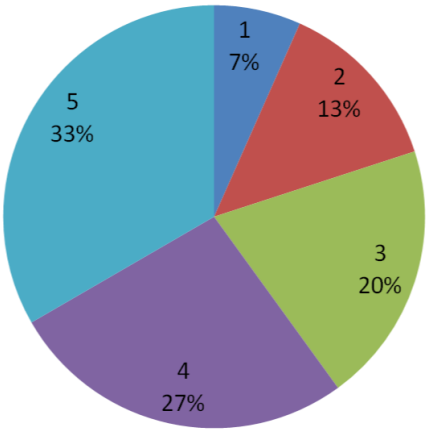
  Pritisak predstavlja odnos verovatnoća preživljavanja dobrih i loših jedinki te on može biti veći kada se sa velikom verovatnoćom prenose **bolje jedinke** što dovodi do prerane konvergencije.

  Raznolikost omogućava lošijim jedinkama da budu izabrane.



Rulet selekcija

Rulet selekcija spada u proporcionalne selekcije i pomoću nje se određuje verovatnoća da će jedinka biti selektovana proporcionalno njenom stepenu prilagođenosti.



Individual	Fitness
1	1.0
2	2.0
3	3.0
4	4.0
5	5.0

Čest problem same ruletne selekcije je da se veoma brzo dostigne **prerana konvergencija** zbog dominacije najbolje jedinke, te problem dominacije rešavamo **rangiranjem**.

Stohastička univerzalna selekcija

Stohastička univerzalna selekcija slabijim jedinkama daje šansu da budu izabrane time smanjuje nefer prirodu ruletne selekcije.

K-turnirska selekcija

Sprovodi se više turnira među nekoliko jedinki izabranih slučajno iz populacije. Na turniru se od K slučajno odabranih jedinki pronalazi najbolja.

Najbolja jedinka je pobednik turnira ako ona ima najbolju vrednost fitness funkcije.

Bitno je napomenuti da N turnira daje N roditelja.

Jednostavna turnirska selekcija

Jednostavna turnirska selekcija je specifičan oblik binarne turnirske selekcije za K = 2.

Slučajnim postupkom se bira dva para jedinki, pri čemu se u svakom paru **lošije jedinke eliminišu**.

Ukrštanjem boljih jedinki generiše se dvoje potomaka, koji se potom mutiraju i evaluiraju. Taj par novostvorenih jedinki nadomešćuje eliminisane jedinke.

Na taj način genetski algoritam sa jednostavnom turnirskom selekcijom u istom koraku obavlja

i selekciju i reprodukciju.

Linearno sortirajuća selekcija

Verovatnoća izbora zavisi od položaja jedinke u poretku jedinki sortiranih po fitnesu.

Kod linearno sortirajuće selekcije verovatnoća je proporcionalna rangu, odnosno poziciji jedinke u poretku sortiranom po pogodnosti, pri čemu najbolja jedinka ima indeks **N**, a najgora indeks **1**.

$$p(i) = \frac{i}{\sum_{i=1}^{N} i} = \frac{2i}{N(N+1)}, \qquad i = 1, \dots, N$$

Selekcija (μ+ λ)

1. Slučajno se bira **μ** roditelja.
2. Njihovim ukrštanjem se stvara **λ** potomaka.
3. Iz skupa roditelja i potomaka bira se najboljih **μ** jedinki za sledeću generaciju.
4. Postupak se ponavlja sve dok se ne popuni nova generacija sa **N** novih jedinki, odnosno **N/μ** puta.

Selekcija (μ, λ)

1. Slučajno se bira **μ** roditelja.
2. Njihovim ukrštanjem se stvara **λ** potomaka.
3. Potomaka je više od roditelja (**λ ≥ μ**) i **μ** najboljih potomaka se bira za narednu generaciju

6. Genetski algoritam: Ukrštanje. Tipovi i izbor

- **Ukrštanje** se primenjuje na dva hromozoma roditelja i kreira dva potpuno nova hromozoma tj. njihovo potomostvo, koje sadrži kombinovana svojstva roditelja. Primenjuje se na slučajno odabranim lokacijama hromozoma, tzv. tačkama ukrštanja.

Tipovi ukrštanja:

- **ukrštanje u jednoj tački** (kombinacija roditelja od jedne tačke)
- **ukrštanje u dve tačke** (analogno u N-tačaka)
- **uniformno ukrštanje** (svakibit ima 50% šanse za ukrštanje)

Izbor tipa ukrštanja:

- za **veće populacije** koristi se **ukrštanje u jednoj** ili **dve tačke**
- za **manje populacije** koristi se **uniformno ukrštanje**
- za **rutiranje putanja** koristi se **ograničeno ukrštanje u N tačaka**

Karakteristike ukrštanja realnih vrednosti

Prednosti

- Jednostavni i brzi proračuni
- Moguće je generisati veliki broj dece od dva roditelja
- Lako se upravlja širokim spektrom varijacija

Ograničenja

- Potrebe za generisanjem α_i i β_i
- Neiskusni korisnici teško izaberu korektne vrednosti
- Ako α_i i β_i nisu dobri, rešenje se zaglavi u **lokalnom minimumu**

Ukrštanja permutacija

Nasumično se odabere određeni broj indeksa gena. Na tim indeksima se preuzmu geni jednog roditelja i utvrdi se koji su geni iskorišćeni i oni neće biti uzeti u razmatranje u drugom roditelju.

Preostali geni se uzimaju od drugog roditelja u redosledu u kojem su poređani i popunjavaju se s leva na desno na preostala mesta u jedinki koja se formira.

Ukrštanjem dva roditelja nastaje samo jedna jedinka!

Roditelj 1	1 4 7 3 8 2 5 6
Roditelj 2	3 6 2 8 1 7 4 5
Dete	6 8 7 3 1 2 4 5

7. Genetski algoritam: Mutacija. Vrste mutacije

- **Mutacija** je način kreiranja novih individua iz populacije pravljenjem manjih promena gena na slučajnim lokacijma već postojećih hromozoma.

  Kao i u prirodi, mutacije su neželjene u najvećem broju slučajeva, tako da su koeficijenti mutacije uglavnom dosta niski.

  U slučaju **binarnog kodiranja**, osnovna mutacija podrazumeva promenu 0 u 1 i obrnuto.

  Kod **realnih vrednosti** mutaciju ostvarujemo sabiranjem sa malim slučajno generisanim brojem. Kod realno kodiranih jedinki se može koristiti veći stepen mutacije jer se kod njih povećava nivo

moguće pretrage prostora rešenja i ne utiče negativno na konvergenciju.

Vrste mutacija:

- **slučajna mutacija**
- **polinomska mutacija**

Slučajna mutacija koristi pravilo:

$$P_{mutirano} = P_{original} + (r - 0.5) \cdot \Delta$$

gde je $r \in [0,1]$ slučajan broj, a Δ je maksimalna vrednost promene definisana od strane korisnika.

Polinomska mutacija je zasnovana na polinomskoj raspodeli i sastoji se iz sledećih koraka:

- Generisati slučajan broj **r** između 0 i 1
- Izračunati faktor promene **δ**

Mutirano rešenje se dobija kao:

$$P_{mutirano} = P_{original} + \delta \cdot \Delta$$

gde je **Δ** je maksimalna vrednost promene između originalne i mutirane vrednosti

8. Genetski algoritam: Značaj i varijacije operatora mutacije

- **Mutacija** je važna zbog izbegavanja lokalnih optimuma. Dovoljno je da jedna jedinka (nastala mutacijom) bude bolja od ostalih, pa da se u nekoliko narednih generacija, sve jedinke presele u prostor gde se nalazi bolje rešenje.

  Varijacije operatora mutacije:

  - **naklonjenost mutacije jedinkama sa manjim fitnesom** da bi se proširilo polje pretrage, a istovremeno očuvale informacije kod onih sa većim fitnesom

  - **parametrizacija mutacije** tako da frekvencija mutacije opada sa konvergencijom populacije.

9. Genetski algoritam: Rekombinacija. Vrste

- **Rekombinacija** je proces kombinovanja genetskih operatora ukrštanja i mutacije pri čemu potomci imaju mnoge karakteristike roditelja.

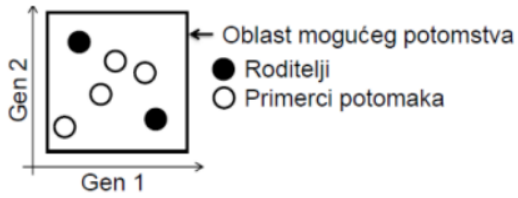
  Vrste rekombinacije:

  - **intermedijalnom rekombinacijom** dobijaju se nove jedinke okolo i između roditelja. α se bira za svaki par roditeljskih gena $\alpha \in [-0.25, 1.25]$

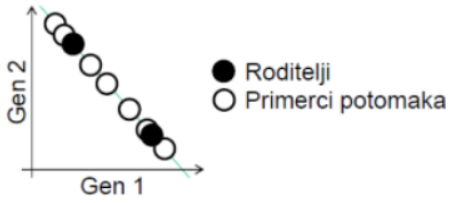
    $$Q_1 = \alpha_1 \times P_1 + (1 - \alpha_2) \times P_2$$

  - **linearna rekombinacija** α je jedna vrednost u toku rekombinacije

    $$Q_1 = \alpha \times P_1 + (1 - \alpha) \times P_2$$



Intermedijalna rekombinacija



Linearna rekombinacija

10. Genetski algoritam: Elitizam i kriterijumi zaustavljanja

- Dobro rešenje dobijeno nakon puno iteracija može se izgubiti izmenom od strane genetskih operatora mutacije ili selekcije.

  Zbog toga se javlja potreba da se najbolje jedinke zaštite od izmene ili eliminacije tokom evolutivnog procesa. Takav mehanizam se naziva **elitizam**.

  Genetski algoritam sa ugrađenim elitizmom iz generacije u generaciju asimptotski teži ka rešenju problema - globalnom ekstremu.

  Međutim, da bi se u svakom koraku evolucije zaštitila najbolja jedinka od izmena ili eliminacije, potrebno ju je prethodno pronaći.

  Pretraživanje ili sortiranje zahteva dodatno procesorsko vreme zbog čega se može znatno usporiti genetski algoritam.

Kriterijum zaustavljanja:

- dostignut je **zadati broj iteracija**
- dostignuta je **željena vrednost kriterijumske funkcije**
- dostignuto je **vreme trajanja algoritma**

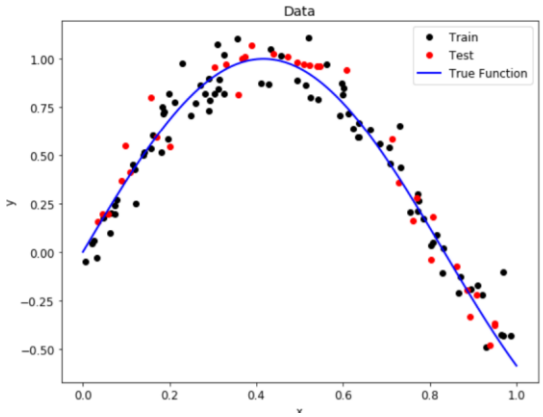
Zaglavljivanje algoritma se dešava ako se prilagođenost najbolje jedinke nije značajno promenila u zadatom broju generacija, te se rešava blagom korekcijom parametara npr. povećanjem stepena mutacije.

11. Izbor modela: Unakrsna validacija i vrste

- **Izbor modela**: Neka je model predstavljen u obliku:

$$h_{\theta(x)} = \theta_0 + \theta_1 x + \theta_2 x^2 + \cdots + \theta_k x^k$$

Potrebno je odrediti
$k \in \{0,1,2, \dots , 10\}$.



Zadatak je odabrati model iz skupa modela:

$$\mathcal{M} = \{ M_1, M_2, \dots , M_k \}$$

Unakrsna validacija (Cross validation)

Trenirati svaki model M_i na obučavajućem skupu S tako da se dobiju se hipoteze h_i.

Odabir hipoteze sa najmanjom greškom

Odabir ovakve hipoteze neće raditi jer ako se izabere polinom velikog reda on će bolje fitovati podatke iz obučavajućeg skupa S i dati manju obučavajuću grešku ali će onda dati veliku varijansu što nije dobro.

Algoritmi validacije:

- Jednostavna unakrsna validacija
- K-tostruka validacija
- Validacija jednostruke eliminacije

Jednostavna unakrsna validacija

1. Na slučajan način se podeli S na **obučavajući skup** S_{train} i i na **validacioni skup** S_{cv}
2. Trenira se svaki model na skupu S_{train} i dobijaju hipoteze h_i

3. Bira se hipoteza $\mathbf{h}_i$ sa najmanjom greškom $\varepsilon_{SCV}(h_i)$ na validacionom skupu S_{cv}

Nedostatak unakrsne validacije je da se izgubi oko ⅓ podataka!

12. Izbor svojstava: Algoritmi

- **Izbor svojstava:** Ako je broj svojstava d veoma velik $d \gg n$ samo je mali skup svojstava relevantan.

  Za veliki broj podskupova pretraga postaje **obimna**.

  Algoritmi omotača
  - Pretraga unapred – dodavanje svojstava
  - Pretraga unazad – uklanjanje svojstava

  Rangirajući algoritmi
  - Uzajamna informacija svojstva i izlaza - korelacija

  Algortimi omotača – pretraga unapred
  1. Polazi se od praznog skupa svojstava $x = \emptyset$
  2. Odlika $x_j \notin x$ sa najmanjom greškom se dodaje u $x = x \cup x_j$
  3. Ako postoji još neizabranih odlika => korak 1
  4. U suprotnom vrati podskup x, kraj

  Algortimi omotača – pretraga unazad
  1. Polazi se od skupa svih svojstava x
  2. Eliminiše se u svakoj iteraciji jedno svojstvo x_j sa najvećom greškom $x = x \setminus x_j$
  3. Ako postoji još neizabranih odlika => korak 1
  4. U suprotnom vrati podskup x, kraj

13. PSO algoritam: Osnovni principi

- PSO algoritam je **evolutivna** odnosno **populaciona** tehnika za koju se posmatra se skup tačaka koje nazivamo populacija ili roj.

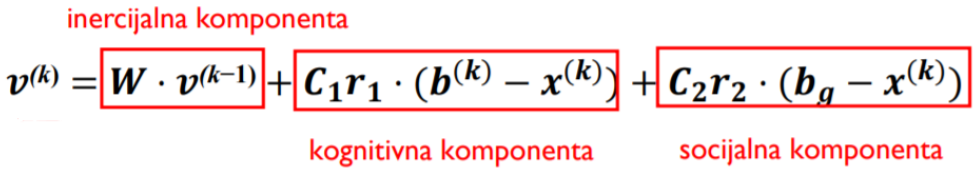
  Tačke se nazivaju **jedinkama** ili **česticama**.

  Jedinka se pomera unutar prostora pretrage, stoga se za PSO ponekad kaže da je **swarm-based** optimizaciona tehnika.

Svakoj jedinki se sem **tekuće pozicije x** pridružuje i **tekuća brzina v.**

Svaka jedinka je u stanju da pamti svoju najbolju ikada dostignutu poziciju b_9.

Roj pamti najbolju poziciju ikada dostignutu od strane bilo koje svoje čestice b_9.



W – faktor inercije
C_1 - faktor individualnosti (konstanta) ili kognitivni faktor
C_2 - socijalni faktor (konstanta)
$b_i - x_i^{(k-1)}$ - individualna komponenta (b_i je najbolja pozicija jedinke)
$b_9 - x_i^{(k-1)}$ - socijalna komponenta (b_g - najbolja pozicija svih čestica)
r_1 i r_2 - slučajni brojevi iz intervala [0,1]
x_i^k – pozicija jedinke i u k-toj iteraciji
v_i^k – brzina kretanja jedinke i u k-toj iteraciji

14. PSO algoritam: Modifikacije PSO algoritma

Faktor inercije

U osnovnoj varijanti PSO algoritma faktor inercije je **W = 1.**

Neophodno je veštački ograničiti maksimalnu brzinu svake čestice.

Uočeno je da se performanse algoritma bitno poboljšavaju uvođenjem promenljivog inercionog faktora [0.4, 0.9].

Ograničenje brzine

Pri velikim brzinama kretanja jedinke dešavaju se **značajne promene pozicije** što dovodi do **divergencije roja.**

Kako bi se izbegla divergencija roja postavlja se ograničenje **maksimalno dozvoljene brzine v_M:**

$$v_{Mi} = k \cdot (b_i - a_i)$$

U cilju raznovrsnije brzine kretanja za sve dimenzije jedinke, dodatno:

$$v_i^{(k+1)} = v_i^{(k+1)} \cdot \Delta x_{max} \cdot r$$

Δx_{max} - maksimalno moguća promena pozicije
r - slučajan broj $r \in [0, 1]$

Koncept okoline

Koncept okoline b_{gj} uvodimo kako bi izbegli zaglavljivanje u lokalnom optimumu zbog usmeravanja svih čestica ka najboljoj globalnoj poziciji b_g.

Socijalni i kognitivni faktor

Promenljivi socijalni i kognitivni faktor mogu poboljšati performanse algoritma.

Promenljivi faktori ubrzanja (C_1 i C_2) povećavaju sposobnost algoritma da pretražuje prostor pretrage i smanjuju verovatnoću zaglavljivanja u lokalnom optimumu.

15. Diskretni PSO algoritam

- PSO je algoritam za rešavanje kontinuiranih problema.

Varijante algoritma za diskretne sisteme

- binarni PSO (svaka jedinka iz populacije može uzeti binarne vrednosti)
- celobrojni PSO (zaokruživanje dostignutog realnog optimuma na najbliži ceo broj ili se u okolini dobijenog rešenja traži najbolje celobrojno rešenje)

Binarni PSO

Umesto:

$$x_i^{k+1} = x_i^k + v_i^k$$

Pozicija čestice se određuje kao:

$$x_{i,d}^{(k+1)} = \begin{cases} 1, & rand() < S(v_i^{(k)}) \\ 0 \end{cases}$$

gde je **d** indeks bita čestice, **S** je **sigmoidna funkcija**:

$$S(x) = \frac{1}{1 - e^{-x}} = \frac{e^x}{e^x + 1}$$

Celobrojni PSO

Neophodno je prilagoditi PSO diskretnom problemu tj. definisati dozvoljeni prostor za pozicioniranje čestica.

Brzina se zaokružuje na najbliži ceo broj.

$$v_i^{(k+1)} = round(v_i^{(k+1)})$$

Pozicija čestice mora imati celobrojne vrednosti iz nekog intervala.

$$x_i^{(k+1)} = |\mathrm{mod}(x_i^{(k+1)}, m)|$$

gde je **mod(ρ, q)** – ostatak prilikom deljenja **ρ** sa **q**.

16. Distribuirani PSO algoritam

- Zasniva se na paralelnom pretraživanju prostora dopustivih rešenja i ima analogije sa konceptom okoline. Takođe se može posmatrati kroz više nivoa pretrage.

Populacija se sastoji iz više distribuiranih podrojeva.

Princip

1. PSO nad svakim lokalnim podrojem
2. Razmena rezultata
3. Ponavljanje koraka 1. i 2. do završetka algoritma

Sinhronizacija

Period sinhronizacije (T_{sync}), određuje frekvenciju komunikacije između podrojeva, broj iteracija posle kojeg će se ažurirati globalni optimum (b_g). Kada podrojevi dobiju informaciju o globalnom optimumu tada menjaju brzinu.

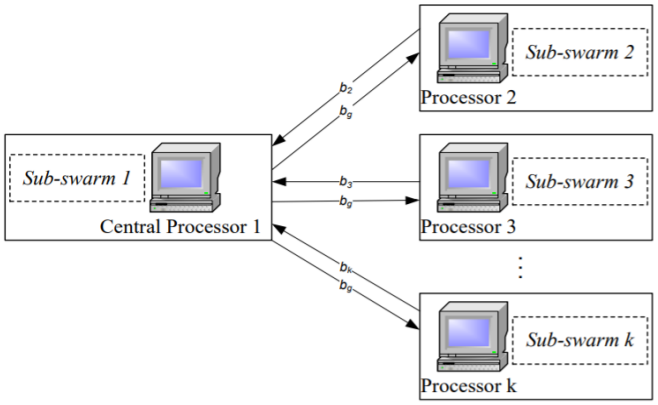
Promenom perioda sinhronizacije se menja i način izvršavanja algoritma.

Malo T_{sync} - podrojevi su više međusobno zavisni i efekat paralelizma manje dolazi do izražaja.

Kada koristimo malo T_{sync} algoritam je sporiji zbog češće komunikacije između podrojeva.

Kada koristimo **veće** T_{sync} podrojevi su nezavisniji, ali se to može odraziti na kvalitet rešenja.

Arhiktetura sistema



17. Evolutivni PSO algoritam

- **Hibridizacija algoritama** se vrši u cilju poboljšanja efikasnosti nekog algoritma koji je korišćen za globalno pretraživanje kako bi brže došli do zadovoljavajućeg i boljeg rešenja.

  EPSO algoritam podstaknut je idejom da se kreira algoritam koji bi bio kombinacija PSO algoritma i evolutivnih algoritama, pri čemu bi se umesto klasične mutacije i rekombinacije koristilo pravilo kretanja čestica koje je definisano PSO algoritmom.

  Faze EPSO algoritma

  1. **Replikacija** – svaka čestica se replicira r puta
  2. **Mutacija** – svaka čestica ima mutirane strateške parametre
  3. **Reprodukcija** – mutirana čestica stvara potomka na osnovu kretanja čestice zasnovanog na PSO algoritmu
  4. **Evaluacija** – računanje kriterijumske funkcije za svakog potomka
  5. **Selekcija** – na osnovu izabrane procedure (slučajnom, turnirskom, proporcionalnom) najbolje čestice opstaju i prenose se na sledeću generaciju

Pravilo kretanja EPSO

$$x_i^{k+1} + x^k + v_i^{k+1}$$

$$v_i^{k+1} = w^*_{i1}\, v_i^k + w^*_{i2}\,(b_i - x_i) + w^*_{i3}\,(b_9^* - x_i)\; P$$

b_i – najbolje rešenje na nivou čestice
b^*_9 - mutirano najbolje rešenje na nivou roja
w^*_{i1}, w^*_{i2}, w^*_{i3} - mutirani strategijski parametri
τ, τ' - parametri koji utiču na stepen mutacije
P - stepen komunikacije

Koncepti komunikacije između čestica

U **PSO** komunikacija se odvija u mreži **zvezde**, dok kod **EPSO** algoritma komunikacija se odvija preko **stohastičke zvezde**.

Komunikacija u EPSO algoritmu zasnova je na **stepenu komunikacije P (nije fiksna** i **jedinstvena**) koji predstavlja verovatnoću uticaja globalno najbolje čestice na pojedinačnu česticu.

18. ACO algoritam: Osnovni principi

- Proučavanja kolektivnog ponašanja insekata omogućila su da se razviju moćne metode i algoritmi za distribuirano upravljanje i optimizaciju. Kretanje mrava i pčela inspirisali su razvoj **kolektivne inteligencije**. Mravi uvek biraju najkraći put do hrane.

Razvijeni su **mravlji algoritmi** i **sistemi** koji su se pokazali uspešni pri rešavanju i najsloženijih kombinatornih problema.

Mravlji algoritam (Ant Colony Algorithm – **ACO**) je optimizacioni algoritam koji imitira ponašanje mravlje kolonije i predstavlja jedan od najefikasnijih algoritama za pronalaženje približnog rešenja problema trgovačkih putnika.

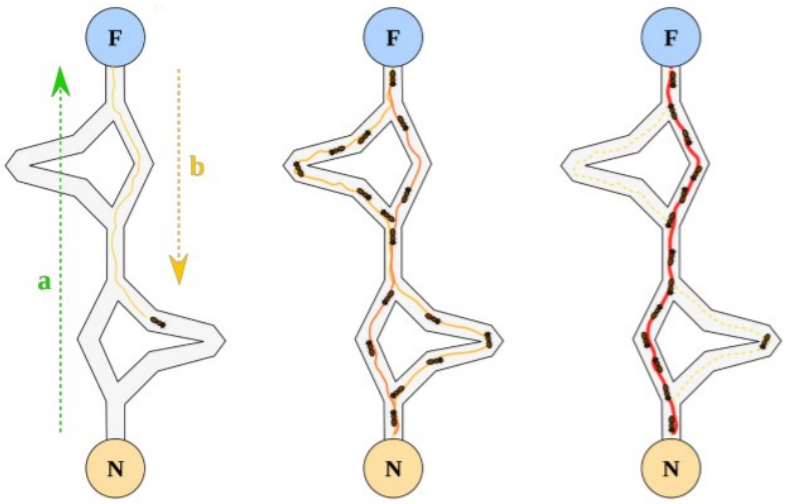
Eksperiment sa preprekom

Glavna ulogu kod mrava ima hemijska supstanca - **feromon**. Mravi nakon pronalaska hrane ostavljaju za sobom **feromonski trag**.

Trag predstavlja signal ostalim mravima da je hrana pronađena pri čemu se svaki mrav kreće nasumično i svi mravi osećaju feromon.

Feromoni s vremenom isparavaju.

Mravi preferiraju praćenje putanje na kojoj postoji deblji sloj feromona pri čemu se prilagođavaju promenama u okruženju.



Algoritam

1 Inicijalizacija tragova feromona, postavljanje parametara
2 **while** nije zadovoljen kriterijum zaustavljanja **do**
3 **for** svakoga mrava **do**
4 Konstrukcija rešenja koristeći trag feromona
5 **end for**
6 Ažuriranje feromona: Isparavanje i Pojačavanje;
7 **end while**
8 Najbolje rešenja ili skup rešenja.

Osnovna verzija ACO algoritma

Atraktivnost grane (i, j) A_{ij}

$$A_{ij} = \tau^{\alpha}_{ij} \cdot \eta^{\beta}_{ij}$$

Verovatnoća prelaska mrava preko grane i-j - p_{ij}

$$p_{ij} = \frac{A_{ij}}{\sum_{k \in S} A_{ik}}, \qquad \forall j \in S$$

gde je S skup suseda čvora i.

Izbor grane na osnovu verovatnoća ρ_{ij} i principom Ruletnog točka.

Mravi ostavljaju trag nakon uspešnog pređenog puta.

Količina feromona zavisi od vrednosti i kvaliteta nađene putanje.

19. ACO algoritam: Promena količine feromona – strategije

- Nakon što svi mravi generišu putanje feromoni se koriguju.

1. **Smanjenje – Evaporacija** feromona se primenjuje po:

$$\tau_{ij} = (1 - \rho)\,\tau_{ij}\,, \qquad \forall (i, j) \in G$$

gde je $0 < \rho \leq 1$ – koeficijent isparenja feromona.

Ciljevi su izbegavanje preuranjene konvergencije u lokalnom optimimu i podsticanje raznovrsnosti pretraživanja.

2. **Povećanje – Pojačanje** feromona zavisi od problema koji ACO rešava

1. **Online - u svakom koraku =>** trag feromona τ_{ij} se osvežava od strane svakog mrava u svakom koraku konstrukcije rešenja.

2. **Online – sa zadrškom =>** trag feromona τ_{ij} se osvežava tek kada mrav generiše kompletno rešenje. Nakon toga mrav osvežava feromonski trag proporcionalno kvalitetu dobijenog rešenja.

3. **Offline varijante =>** trag feromona τ_{ij} se osvežava tek kada svi mravi generišu kompletno rešenje.

Ova strategija se najčešće koristi u različitim vidovima:

a) Na osnovu kvaliteta rešenja => trag feromona se osvežava vrednošću koja je proporcionalna najboljem pronađenom rešenju.

b) Na osnovu rangiranja => samo mravi koji su našli **w** najboljih rešenja mogu da osveže trag feromona, u skladu sa rangom kvaliteta rešenja.

c) Korekcija najgoreg rešenja => mravi koji generišu najgore rešenje će smanjiti trag feromona.

d) Elitizam => samo mrav koji je našao najbolje rešenje će pojačati feromon da bi usmerio pretragu u tom smeru.

20. Mašinsko učenje: Definicija i osnovna podela

- **Mašinsko učenje** temelji se na algoritmima koji mogu učiti iz podataka bez oslanjanja na programiranje zasnovano na pravilima.

Podela mašinskog učenja

- Nadgledano učenje
- Nenadgledano učenje
- Polunadgledano učenje
- Učenje s podsticajem

Nadgledano učenje

Polazi od skupa parova (vektora svojstava odnosno obeležja)

Cilj je na osnovu ulaznih podataka - parova izvesti pravilo koje predviđa vrednost asociranu sa novim vektorom svojstava:

$$\mathbf{x} \rightarrow \mathbf{y}$$

Regresivni modeli asociraju realan broj sa svakim vektorom svojstava, dok **klasifikacioni modeli** asociraju jedan simbol iz konačnog skupa sa svakim vektorom svojstava

Algoritmi nadgledanog učenja

- K-Nearest Neighbors
- Linear Regression
- Logistic Regression
- Support Vector Machines (SVMs)
- Decision Trees and Random Forests
- Neural networks

Nenadgledano učenje

Koncept koji koristi neoznačene podatke samo X i nalaze interesantne stvari o njima.

Najčešće se primenjuje za grupisanje podataka npr. google news web site - izdvaja članke na istu temu od različitih autora.

Algoritmi nenadgledanog učenja

- **Grupisanje**
  (K-Means, Hierarchical Cluster Analysis, SVM)

- **Detekcija anomalija**
  (One-class SVM, Isolation Forest)

- **Vizuelizacija i smanjenje dimenzionalnosti**
  (Principal Component Analysis, Kernel PCA, Locally Linear)

- **Učenje pravila udruživanja**
  (Equivalence Class Clustering and b-up Lattice Traversal)

21. Merenje rastojanja između podataka

Euklidsko rastojanje

$$d(A,B) = \sqrt{\sum_{i=1}^{n}(a_i - b_i)^2}$$

Menhetn rastojanje

Suma apsolutnih vrednosti razlika ulaza u vektorima:

$$Manhattan = \sum_{i=1}^{n} |x_i - y_i|$$

Čebiševo rastojanje

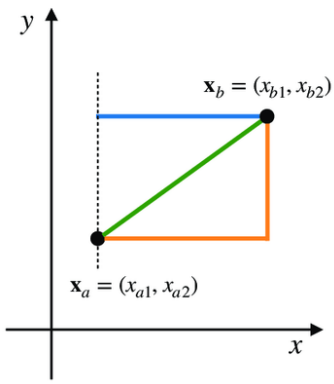
Maksimum abs vrednosti razlika između svih elem. vektora:

$$Chebyshev = max(|x_i - y_i|)$$

Sličnost kosinusa

Vektori usmereni u istom smeru su slični:

$$similarity(a,b) = \frac{a.b}{\|a\|.\|b\|}$$



$p = 2$ Euclidean distance

$\|\mathbf{x}_a - \mathbf{x}_b\|_2 = (|x_{a1} - x_{b1}|^2 + |x_{a2} - x_{b2}|^2)^{\frac{1}{2}}$

$p = 1$ Manhattan distance

$\|\mathbf{x}_a - \mathbf{x}_b\|_M = |x_{a1} - x_{b1}| + |x_{a2} - x_{b2}|$

$p = \infty$ Chebyshev distance

$\|\mathbf{x}_a - \mathbf{x}_b\|_\infty = \max\{|x_{a1} - x_{b1}|, |x_{a2} - x_{b2}|\}$

22. Skaliranje podataka

- **Skaliranje** je prvi korak u cilju pravilnog poređena svojstava kod merenja rastojanja između tačaka. Ako se ljudi klasifikuju prema plati i broju kućnih ljubimaca, vrednosti za platu su preko 30000, a vrednosti za broj kućnih ljubimaca je < 10.

  Jedan od načina je da se po svakom svojstvu u svim podacima odrede srednja vrednost i da se skaliraju vrednosti tog svojstva tako da imaju srednju vrednost 0. Drugi način je da je minimalna vrednost 0, a najviša 1

23. K-means klasterizacija: Izbor optimalnog k

- **K-means klasterizacija** predstavlja nenadgledanu tehniku učenja koja grupiše veliki broj tačaka predstavljenih vektorima koje nisu ni klasifikovane ni označene, pri čemu je svaka grupa asocirana svojim centroidom odnosno težištem.

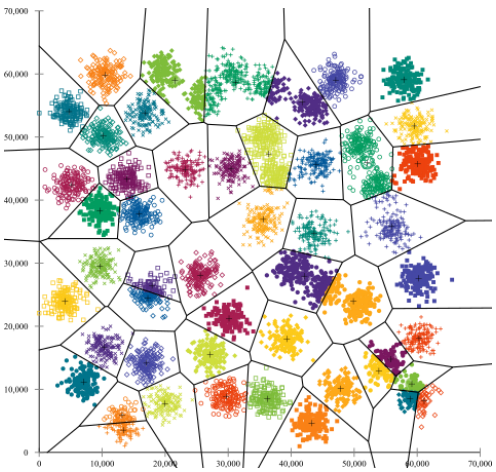
  Vrste problema klasifikacije:

  1. Traženje strukture unutar skupova neklasifikovanih podataka, sa pretpostavkom o kategorijama

  2. Veštačko deljenje podataka - ne postoji očigl. grupisanje

  K-means algoritam

  K-MEANS (X, K)
  1 inicijalizacija: slučajan izbor K težišta $\mu_1, \mu_2, \dots, \mu_K \in \mathbb{R}^n$
  2 **repeat** (do konvergencije težišta)
  3 **for** i = 1 **to** m
  4 $c^{(i)}$ = ind. klast. sa težišt. najb. $x^{(i)}$
  5 **for** k = 1 **to** K
  6 μ_k = težište za tačke iz klastera k
  7 **return** $C = \{c_{(1)}, c_{(2)}, \dots, c_{(m)}\}$



Višest. sluč. inic. – izbor optimuma
for i = 1 **to** 100
Sluč. inicijaliz. K-means
Izvrši K-means
Izrač. $J(c_1, \dots, c_m, \mu_1, \dots, \mu_k)$

Izabrati podelu koja daje inimalno:
$J(c_1, \dots, c_m, \mu_1, \dots, \mu_k)$

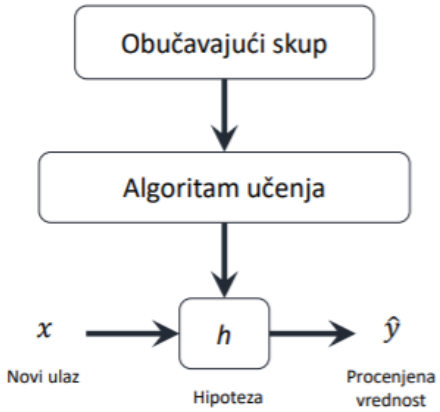
24. Linearna regresija

- **Linearna regresija** se odnosi na svaki pristup modelovanja relacija između jedne ili više zavisnih promenljivih označenih sa Y, i jedne ili više nezavisnih promenljivih označenih sa X, na način da takav model linearno zavisi od nepoznatih parametara procenjenih iz podataka.

  Model

  $$y_i = b_0 + b_1 x_i + e_i$$

  Sam model ulazne podatke deli na dva skupa: **obučavajući skup** i **skup za testiranje**.



  Dve glavne vrste upotrebe linearne regresije su:

  - **Predviđanje vrednosti izlaza y** za nove podatke na osnovu poznatih vrednosti tog izlaza na starim podacima

  - **Analiza stepena povezanosti** između vrednosti izlaza y i vrednosti promenljivih $x_1, x_2, \ldots, x_n$

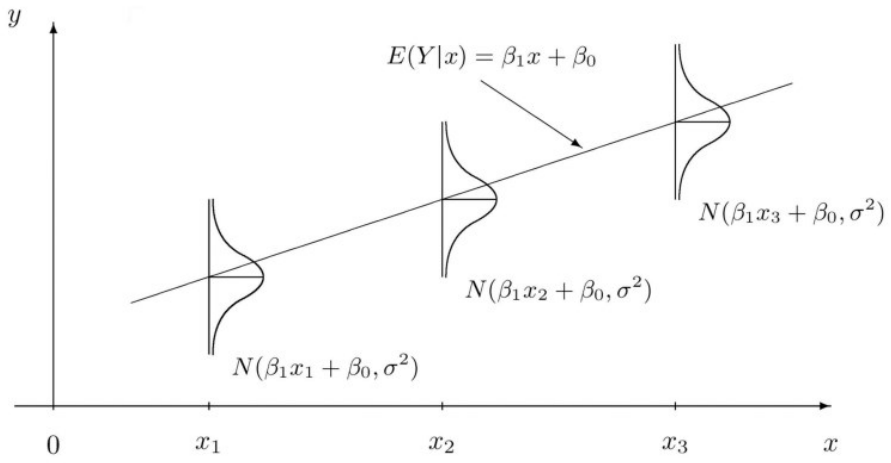
  Normalna raspodela konstantne standardne devijacije sa linearnim modelom proseka

  Ovakav model prikazuje raspodelu sa prosekom koja zavisi od neke promenljive. Najčešće se podrazumeva da je skup atributa proširen atributom koji je konstantne vrednosti 1, kako bi model sadržao slobodni koeficijent.

  Ukupni model je opisan na sledeći način:

  $$p_w(y|x) = \mathcal{N}(w \cdot x, \sigma^2)$$

Kako modeluje uslovnu raspodelu, ovaj model je očigledno probabilistički diskriminativni.



Funkcija gubitka

Funkcija gubitka L(h(x), y) definiše meru odstupanja vrednosti hipoteze h(x) od tačne vrednosti y na pojedinačnom podatku.

Funkcija greške

Funkcija greške J(w) je prosek vrednosti funkcije gubitka iz posmatranog skupa:

$$J(w) = \frac{1}{m}\sum_{i=1}^{m} L(h(x^{(i)}), y^{(i)})$$

25. Vrste gradijentnog algoritma

- **Paketni** ili **šaržni =>** svi podaci se uzimaju odjednom u obzir. Nedostatak ovog algoritma jeste to što za veliku količinu podataka svi se moraju učitati iz baze i sporije dolazi do rešenja.

- **Stohastički =>** izvršava se u svakom koraku za npr. jedan slučajan podatak. Npr. Za cenu kuće, 1. korak za kuću 1, 2. korak za kuću 2. Za veliko m je brži od Paketnog gradijentnog algoritma.

26. Lokalno ponderisana linearna regresija

- Fituje se θ za minimizaciju za svaki od primera, pri čemu je w^i težinska funkcija.

Ako je $|x^i - x|$ veliko, $w^{(i)} \approx 0$ tada je mali uticaj na J.

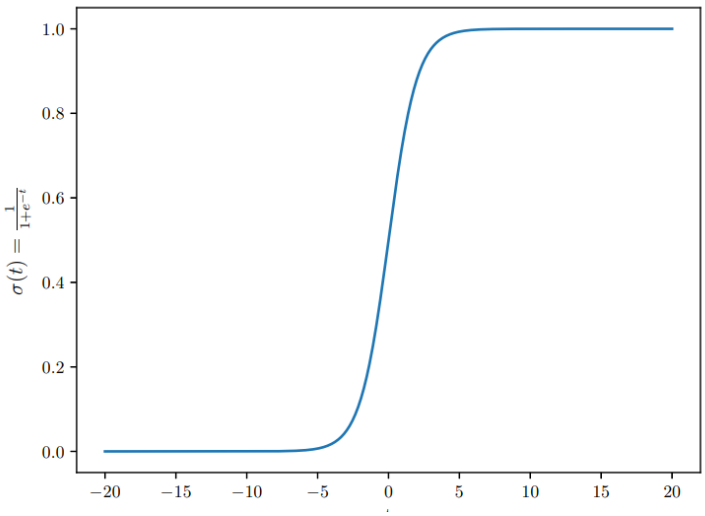
Ako je $|x^i - x|$ malo, $w^{(i)} \approx 1$:

$$J(\theta) = \sum_{i=1}^{m} w^{(i)}(y^{(i)} - \theta^T x^{(i)})^2$$

27. Logistička regresija: Postupak određivanja parametara

- **Logistička regresija** predstavlja jedan od najpopularnijih algoritama klasifikacije i spada u **diskriminativne modele**.

Osnovna verzija algoritma služi za **binarnu klasifikaciju** i kao takva predstavlja sastavni element mnogih arhiktektura neuralnih mreža.



Ime je dobila po logističkoj tj. **sigmoidnoj funkciji** koja se koristi u hipotezi i ima oblik:

$$g(z) = \frac{1}{1 + e^{-z}}$$

Logistička funkcija sabija interval (-∞, +∞) na opseg [0, 1].

Hipoteza logističke regresije je:

$$h(x) = \frac{1}{1 + e^{-(w_0 + w_1 x_1 + w_2 x_2 + \cdots + w_n x_n)}}$$

Funkcija greške

Za odabir funkcije greške biramo onu funkciju koja je konveksna uz preduslov da je i **funkcija gubitka konveksna**.

Kvadrat odstupanja h(x) od y nije pogodna funkcija gubitka za logističku regresiju, jer je zbog logističke funkcije u okviru hipoteze takva funkcija gubitka **nekonveksna**.

Pored konveksnosti, funkcija gubitka bi trebala da ispoljava sledeće ponašanje:

- Kada je **y = 1** potrebno je da greška bude velika kada vrednost $h_{(x)}$ blizu 0, a mala kada je vrednost $h_{(x)}$ blizu 1

- Kada je **y = 0** potrebno je da greška bude velika kada je vrednost $h_{(x)}$ blizu 1, mala kada je vrednost $h_{(x)}$ blizu 0

Postupak određivanja parametara

1. Pretpostavlja se probabilistički model:

$$P(y = 1 \mid \boldsymbol{x};\ \boldsymbol{\theta}) = h_\theta(\boldsymbol{x})$$
$$P(y = 0 \mid \boldsymbol{x};\ \boldsymbol{\theta}) = 1 - h_\theta(\boldsymbol{x})$$
$$p(y \mid \boldsymbol{x};\ \boldsymbol{\theta}) = (h_\theta(\boldsymbol{x}))^y (1 - h_\theta(\boldsymbol{x}))^{1-y}, \qquad y \in [0,1]$$

2. Primenjuje se princip maksimalne verodostojnosti za ceo obučavajući skup

$$\mathcal{L}(\theta) = p(\boldsymbol{y} \mid X;\ \boldsymbol{\theta}) = \prod_{i=1}^{m} p\left(y^{(i)} \middle| \boldsymbol{x^{(i)}};\ \boldsymbol{\theta}\right) = \prod_{i=1}^{m} \left(h_\theta\left(\boldsymbol{x^{(i)}}\right)\right)^{y^{(i)}} \left(1 - h_\theta\left(\boldsymbol{x^{(i)}}\right)\right)^{1-y^{(i)}}$$

Primenom Paketnog gradijentnog postupka dobije se da je:

$$\theta_j = \theta_j + \alpha \sum_{i=1}^{m} \left(y^{(i)} - h_\theta\left(x^{(i)}\right)\right) x_j^{(i)} \quad i = 1, \dots, n$$

28. K najbližih suseda: Izbor optimalnog k

- **KNN algoritam** baziran je na supervizorskoj tehnici. Podaci su predstavljeni vektorom obeležja X.

  KNN algoritam ne koristi stvarno učenje i često se naziva **lenjim algoritmom** te je pogodan kod stalne pojave novih podataka.

  Mana KNN je to što klasifikacija novih tačaka može biti veoma spora i podaci neravnomerno raspoređeni.

  Klasifikacija - obojene tačke u prostoru, svaka dimenzija odgovara jednoj osobini, a boja odgovara kategoriji. Cilj je klasifikovati novu tačku ako joj je zadato mesto u prostoru.

Kako radi?

- Počinje sa m klasifikovanih podataka – supervizorsko učenje
- Svaka tačka ima n obeležja
- Kada se pojavi nova tačka potraži se K najbližih tačaka i odredi njena klasa

Algoritam:

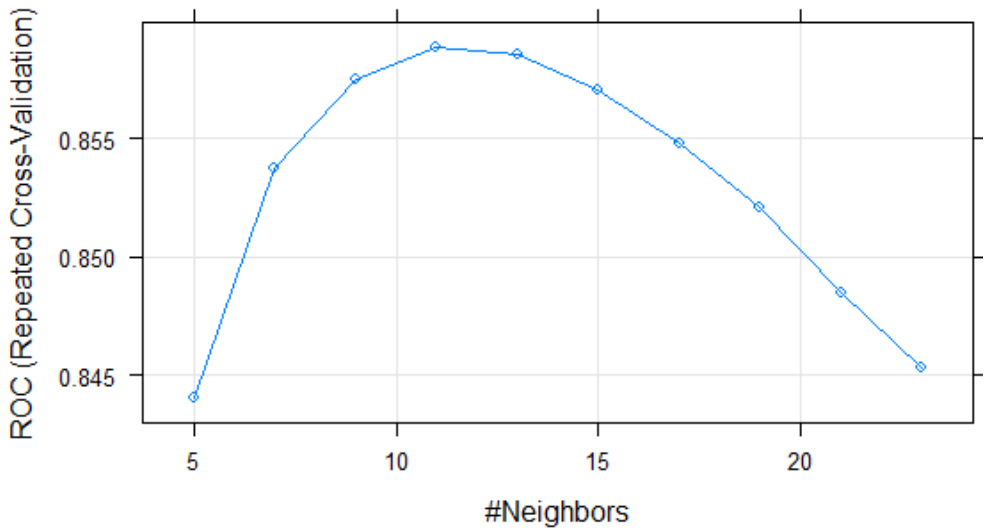
- **Skaliranje** je prilagođavanje atributa da budu uporedivi.
- **Nova tačka** je merenje udaljenosti nove tačke do svih ostalih tačaka. Posmatra se K tačaka sa najmanjim rastojanjem – najzastupljenija klasa među njima je i klasa nove tačke.

Kako odabrati K

Odabir vrednosti za K se najčešće radi pomoću **unakrsne validacije**.

Podelom podataka na trenirajući i validirajući skup npr. 10 puta nekom od **Fold Validation** algoritama, računamo prosečni stepen greške za svaku podelu.

Za vrednost K uzimamo onu vrednost za koju se u toku iteracija izračunao **najmanji stepen greške**.



Malo K daje malu pristrasnost, ali veliku varijansu, dok veliko K daje veliku pristrasnost, ali malu varijansu!

29. Kriptografija (primena, procesi, šifrovanje i bezbednosne pretnje)

- U bitnim programskim sistemima podaci i servisi se moraju očuvati od bezbednosnih pretnji. Bezbednost u računarskim sistemima je usko povezana sa pojmom oslonjivosti.

Šifrovanje (enkripcija)

- **Šifrovanje** je proces u kojem predajna strana modifikuje poruku sa ciljem da ona bude nerazumljiva za potencijalne napadače. Rezultat ovog procesa je **šifrovana poruka**.

Dešifrovanje (dekripcija)

- **Dešifrovanje** je proces u kojem se od šifrovane poruke pravi izvorna poruka.

Tipovi pretnji na nivou mreže

– **Presretanje** je kada neautorizovana stranka dobije pristup podacima ili servisima. Ovde spada i ilegalno kopiranje podataka.

– **Prekidanje** je kada servis ili podatak postane nedostupan, neupotrebljiv ili uništen.

– **Modifikacija** je neautorizovana izmena podataka ili izmena servisa, koji više nije u skladu sa specifikacijom.

– **Fabrikacija** je generisanje dodatnih podataka ili aktivnosti koji ne bi postojali u normalnim situacijama.

- **Malver** (malware - Malicious Software) je pretnja

Mehanizmi sprovođenja bezbednosti

- **Šifrovanje** – transformacija informacije tako da je njeno pravo značenje skriveno. Koristi tajne ključeve kao posebno znanje, pri čemu šifrovana poruka treba da spreči akcije uljeza: presretanje, modifikaciju i fabrikaciju.

- **Autentifikacija** – verifikacija identiteta korisnika, klijenata itd.

- **Autorizacija** – provera da li korisnik ili servis ima pravo na izvršavanje određene akcije.

- **Beleženje istorijata aktivnosti** – upisivanje u dnevnike događaja koji je entitet čemu pristupio i na koji način. Ovaj mehanizam ne obezbeđuje direktnu zaštitu od napada, ali omogućava naknadnu analizu bezbednosnih problema.

30. Osnovni algoritmi kriptografije (podela, primitivni i napredni alg.)

- **Primitivan algoritam šifrovanja** - Jednostavna zamena slova je način kodiranja teksta gde se neko slovo zamenjuje drugim, a dekodiranje ponovi zamenu. Ali ovakvo kodiranje nije dovoljno sigurno, te se često u algoritmima kodiranja koristi ekskluzivno ILI.

One-Time Pad algoritam

Posmatra se niz bita koji čine poruku **ρ**. Za slučajno izabran niz bita iste dužine tretiran kao poruka kažemo da je **binarna podloga** pri čemu sa **k** predstavljamo ključ.

Kako bi dešifrovali poruku potrebno je da poruku postavimo na podlogu i tada dobijamo kodiranu poruku:

$$c = \rho \oplus k$$

Operacija **XOR** se primenjuje na svaki par bita k_i i ρ_i. Kodirana poruka je u obliku gde se teško može rekonstruisati neka dodatna informacija o originalnoj poruci.

Bitne karakteristike algoritma su da čini poruku bezbednom i da koristi ključ iste dužine poruke, pri čemu je ponovna upotreba istog ključa rizična.

Šifrovanja pomoću ključeva

Šifrovanje je proces gde se poruka generiše na osnovu originalne poruke i tajnog ključa.

Dešifrovanje je proces koji generiše originalnu poruku na osnovu kodirane poruke i tajnog ključa.

Nemoguće je naći ključ K ako su poznate poruka P i kodirana poruka C.

Simetrična kriptografija

Simetrična kriptografija koristi isti ključ i za šifrovanje i za dešifrovanje. Zbog toga je raznovrsnost, a samim tim i sigurnost algoritama ovakve enkripcije velika.

Bitan faktor je i brzina jer je simetrična enkripcija veoma brza. Pored svih prednosti koje ima na polju sigurnosti i brzine algoritma, postoji i jedan veliki nedostatak, a to je

kako preneti tajni ključ.

Problem je u tome, što ako se tajni ključ presretne, poruka se može pročitati. Zato se ovaj tip enkripcije najčesce koristi prilikom zaštite podataka koje ne delimo sa drugima.

Uslovi savršene tajnosti:

1. Tajni ključ se koristi samo jednom

2. Kriptoanalitičar ima pristup samo kriptogramu

31. RSA algoritam (principi, uključeni algoritmi)

Asimetrična kriptografija

Za razliku od simetrične kriptografije, koja podrazumeva postojanje jednog kjuča, asimetrična kriptografija uvodi postojanje još jednog ključa.

Jedan ključ ostaje skriven od ostalih učesnika i naziva se **privatni ključ**, dok je drugi ključ poznat svim učesnicima i naziva se **javni ključ**.

Privatni ključ se koristi za enkripciju, javni ključ za dekripciju.

Tri osnovne primene asimetrične kriptografije

- Enkripcija
- Razmena ključa
- Digitalni potpis

Upotreba asimetrične kriptografije široko je zastupljena, od HTTPS/SSL protokola koji se koriste na većini veb sajtova, do kriptovaluta gde javni ključevi predstavljaju identitete elektronskih novčanika.

Isti algoritmi se mogu koristiti za više namena. Jednom privatnom ključu odgovara jedan javni ključ i obrnuto. Poznavanjem javnog ključa izuzetno je teško rekonstruisati privatni ključ.

RSA algoritam

RSA algoritam je algoritam šifrovanja **asimetričnim ključem**.

Algoritam se oslanja na broj koji je proizvod dva velika prosta broja. Sam proizvod je dovoljno veliki da niko u razumno dugom vremenu ne može otkriti činioce.

RSA ALGORITHM

1. Izabrati 2 velika (barem 1024 bita svaki) različita prosta broja p i q
2. Izračunati $n = pq$
3. Izračunati $r = (p-1)(q-1)$
4. Izabrati mali broj e tako da su e i r uzajamno prosti
5. Izračunati d kao inverzan element za množenje $ed \bmod_r = 1$
6. RSA javni ključ je tada par $K_E = (e, n)$
7. RSA privatni ključ je $K_D = (d, n)$
8. Funkcije šifrovanja i dešifrovanja su: $E_{KE}\, x = x_e \bmod(n)$ $D_{KD} x = x^d \bmod(n)$

32. Lanac blokova (Blockchain)

- **Blockchain** predstavlja decentralizovanu, distribuiranu i javnu kolekciju podataka koja je logički organizovana u blokove. Svaki blok je logički povezan sa prethodnikom čime se stvara lanac odatle je nastao naziv ove arhitekture.

  Povezanost blokova se ostvaruje pomoću kriptografije - svaki blok sadrži heš vrednost prethodnog bloka.

  Podaci u svakom bloku predstavljaju skup transakcija koje korisnici kreiraju.

  ### Primena Blockchain-a

  - Finansijske organizacije
  - Osiguravajuća društva
  - Zdravstvene organizacije
  - Sajber bezbednost

  ### Osnovni principi

  Blockchain se može posmatrati kao distribuirana knjiga podataka koja nema centralno skladište. Podaci su dostupni svakom čvoru u mreži čime se postiže transparentnost.

Svaka promena je javna i svaki čvor dobija najnovije stanje u mreži.

Dodavanje novih blokova vrše rudari tj. povezani računari koji koriste veliku količinu električne energije prilikom rešavanja kriptografskih zadataka.

Najviše se primenjuje u oblasti kriptovaluta. Najpoznatije su Bitcoin i Ethereum. Takođe, može se koristiti za skladištenje medicinskih nalaza, kreiranje pametnih ugovora, za analizu poslovnih procesa, itd.

Struktura

Logička predstava Blockhain se interpretira kao lanac blokova gde je svaki blok povezan sa svojim prethodnim.

Glavni elementi su:

- **Heš**
  Heš vrednost se dobija pomoću heš funkcije. Inverzna heš funkcija ne postoji. Ulazne vrednosti su bilo kog tipa podatka, a izlazne su najčešće u heksadecimalnom zapisu. Najčešće korišćena heš funkcija je **SHA256 algoritam**.

- **Lanac blokova**
  Transakcije između korisnika u mreži se skladište u blokove koji su međusobno povezani, čime se kreira lanac.

  Blok sadrži:
  - **Veličinu** koja se izražava u bajtovima
  - **Broj transakcija** smeštenih u blok
  - **Transakcije** su podaci smešteni u blok
  - **Zaglavlje** sadrži polja koja definišu blok

  Sadržaj zaglavlja bloka:
  - **Pokazivač na prethodni blok** sadrži heš vrednost prethodnog bloka

  - **Korenska heš vrednost** predstavlja heš vrednost svih transakcija u bloku koja se dobija pomoću Merkle Tree strukture

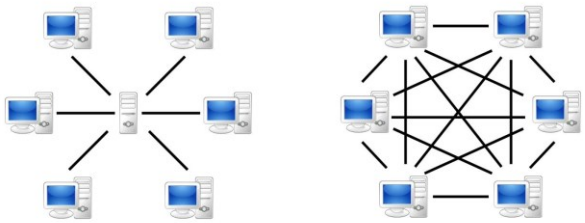
  - **Težina** je parametar koji određuje koliko je vremena i električne energije potrebno da bi se kreirao novi blok

- **Brojač** je jedinstvena, nasumična vrednost koja se može samo jednom iskoristiti za kreiranje bloka. Povećava se ukoliko generisana heš vrednost nije validna za kreiranje novog bloka

- **Vreme** je trenutak kreiranja bloka

- **Verziju** ona određuje validaciona pravila

- **Direktna komunikacija**



- **Digitalni potpis**

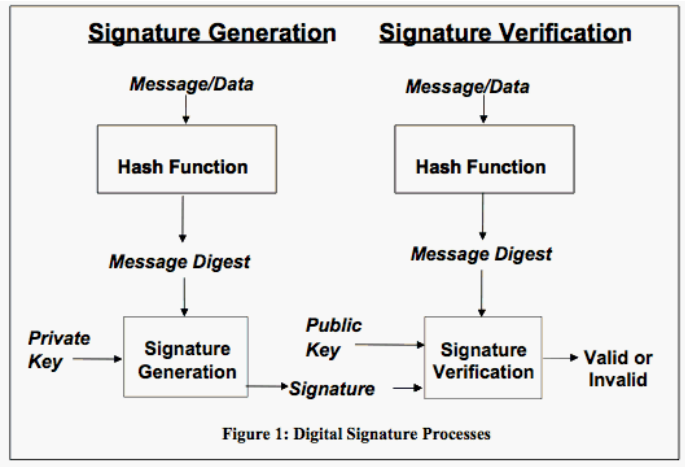


Figure 1: Digital Signature Processes

Algoritmi koncenzusa

Najpoznatiji algoritmi koncenzusa su: Proof of Work, Proof of Stake, Proof of Authority, Practical Byzantine Fault Tolerance, Proof of Burn, Proof of Capacity, Proof of Elapsed Time.

Proof of Work

Najpoznatiji algoritam koncenzusa. Radi po principu: *Rešenje je teško pronaći, ali ga je lako potvrditi.*

U mreži čvorovi koji vrše dodavanje novih blokova se popularno nazivaju rudari. Međusobno se takmiče rešavajući kriptografske zadatke – heš vrednosti sa određenim uslovima.

Rešenje u vidu heš vrednosti predstavlja dokaz o radu, s obzirom da je utrošena značajna količina električne energije.

Dodavanje novog bloka se u proseku izvršava na svakih deset minuta.

Rudari se često udružuju kako bi imali veće šanse prilikom pronalaženja heš vrednosti. Kada je pronađu, nagrada se deli između svih čvorova na osnovu njihove priložene računarske moći.

Proof of Stake

Radi po principu ulaganja kriptovaluta u mrežu.

Validatori su zaduženi za kreiranje blokova. Svaki validator ulaže u mrežu određen iznos u kriptovaluti. Mreža nasumično bira validatora koji će da kreira naredni blok. Onaj sa najvećim ulogom ima najveće šanse da bude izabran. Za kreiranje se dobija nagrada u određenom iznosu kriptovalute.

Prednosti u odnosu na PoW su:

- Energetska efikasnost tj. nije neophodno trošiti resurse i električnu energiju.
- Ušteda u opremi nije neophodno koristiti moćne računare za kreiranje novog bloka.
- Jača otpornost ka centralizaciji znači da je moguće povećati broj čvorova u mreži.
- Podrška za sporedne lance utiče na skalabilnost Ethereum mreže.

Proof of Authority

Obično za odobrene knjige identiteti korisnika moraju biti poznati i verifikovani.
Mogućnost objavljivanja novih blokova je diktirana korisničkim dozvolama. Nema problema u vezi sa procesorskom snagom ili strujom.

33. Dokazi bez otkrivanja informacija (Zero Knowledge Proof)

- **Zero-Knowledge Proof** je protokol dokazuje Viktoru da Pegi ima neke informacije, ali nikako ne otkriva njihov sadržaj.

  Odnos između **Provera** (Pegija) i **Verifiera** (Viktora) je da Viktor postavlja niz pitanja Pegi, pri čemu ako Pegi zna tajnu tačno će odgovoriti na svako od pitanja pri čemu se tajna **ne otkriva**.

Osnovni protokol

1. Viktor stoji u tački A
2. Pegi prelazi ceo put kroz pećinu ili do tačke C ili D
3. Kada Pegi nestane u pećini, Viktor odlazi u tačku B
4. Viktor vikne tražeći od Pegi da:
   a) Izađe iz levog prolaza
   b) izađe iz desnog prolaza
5. Pegi pristaje, koristeći čarobne reči za otvaranje tajnih vrata, ako je to potrebno
6. Pegi i Viktor ponavljaju korake od 1 do 5, n puta

Osnovni protokol se sastoji iz nekoloko faza:

1. Pegi koristi svoju informaciju i slučajno generisani broj kako bi svoj težak problem transformisala u drugi i izomorfni težak problem
2. Pegi predaje rešenje novog primera
3. Pegi otkriva Viktoru novi primer (ne otkriva izvorne info.)
4. Viktor traži od Pegi da:
   a) Dokaže da su stari i novi problem izomorfni.
   b) Otvori rešenje iz koraka 2. i dokaže da je to rešenje novog problema.
5. Pegi pristaje
6. Pegi i Viktor ponavljaju korake 1 - 5, n puta.

Fiat Shamir algoritam

Generiše se broj n kao proizvod dva velika prosta broja.

Za generisanje privatnoj i javnog ključa Pegi bira k različitih brojeva $v_1, v_2, \dots, v_k$ gde je svaki broj v_i kvadratni ostatak po modulu n. Izabere se v_i tako da $x^2 = v_i \ mod(n)$ ima rešenje i da postoji $v^{-1} \ mod(n)$. Niz $v_1, v_2, \dots, v_k$ je javni ključ.

Izračuna se najmanje s_i tako:

$$s_i = \sqrt{v_i^{-1}} \ mod(n)$$

Niz $s_1, s_2, \ldots, s_k$ je privatni ključ

34. Problem vizantijskih generala

- Nekoliko divizija Vizantijske vojske logoruje izvan neprijateljskog grada. Svakom divizijom komanduje njen general. Oni mogu komunicirati jedan sa drugim samo preko glasnika. Nakon posmatranja neprijatelja, moraju se odlučiti za zajednički plan akcije.

  Generali treba da postignu konsenzus o planu. On može biti **povlačenje** ili **napad**.

  Mogu postojati dva tipa generala **lojalni** i **izdajnici**. Svi lojalni generali bi trebali postići **koncenzus**.

  Komandantski general šalje naređenje svojim n-1 general - potpukovnicima tako da:

  1. Svi lojalni generali su poslušni → doslednost/sporazum

  2. Ako je komandujući general lojalan, onda svaki odani general - potpukovnik poštuje naređenje koje šalje komandujući → validnost

  Uslovi koji moraju biti ispunjeni su da svi lojalni generali odlučuju o istom planu akcije i da mali broj izdajnika ne može izazvati loš plan odanih generala.

  Loši rezultati su ako je ⅓ ili više generala izdajice i ne postoji rešenje kada je manje od 3m + 1 generala gde je m broj izdajnika.