1. Šta su to inteligentni sistemi! Navesti definiciju i nekoliko primjera njihove primjene.

Inteligentni sistemi su računarski programi koji koriste umjetnu inteligenciju kako bi rješavali složene probleme i donosili odluke koje bi se inače smatrale ljudskim zadatkom. Ovi sistemi uključuju različite vrste umjetne inteligencije, kao što su strojno učenje, duboko učenje, genetski algoritmi, prirodni jezik obrade i druge tehnologije.

Primjeri inteligentnih sistema uključuju:

1. Osobni asistenti kao što su Siri, Google Assistant i Amazon Alexa koji koriste prirodni jezik obrade kako bi razumjeli korisnikove zahtjeve i odgovorili na njih.

2. Samovozeća vozila koja koriste senzore i algoritme strojnog učenja kako bi navigirala putevima i izbjegavala sudare.

3. Robotski sistemi u proizvodnji koji koriste strojno učenje kako bi optimizirali proizvodne procese i poboljšali kvalitetu proizvoda.

4. Sistemi za preporuke koji koriste algoritme za strojno učenje kako bi analizirali podatke o korisničkim preferencijama i navikama kako bi preporučili proizvode, filmove, muziku i druge sadržaje.

5. Sistemi za prepoznavanje lica i glasa koji se koriste u raznim aplikacijama za sigurnost, poput sustava za prepoznavanje putnika na aerodromima ili u aplikacijama za mobilno bankarstvo.

Inteligentni sistemi sve više pronalaze primjene u raznim industrijskim granama, a očekuje se da će se njihova upotreba u budućnosti nastaviti širiti i unapređivati.

2. Šta je to prostor stanja, a šta veličina prostora stanja?

Prostor stanja je matematički model koji se koristi u oblasti inteligentnih sistema za opisivanje svih mogućih stanja koje sistem može zauzeti tokom izvršavanja nekog zadatka. Konkretno, prostor stanja predstavlja skup svih mogućih kombinacija ulaznih vrijednosti koje mogu uticati na izlaz sistema.

Veličina prostora stanja odnosi se na broj mogućih stanja koje sistem može zauzeti. Što je veći prostor stanja, to je veći broj kombinacija ulaznih vrijednosti koje sistem mora uzeti u obzir kako bi donio odluku ili izvršio neki zadatak. To može dovesti do povećanja složenosti sistema i utjecaja na vrijeme izvršavanja i tačnost rezultata.

U praksi, inteligentni sistemi koriste različite metode za smanjenje veličine prostora stanja, uključujući redukciju dimenzionalnosti podataka, odabir značajki, grupisanje podataka i druge tehnike. Cilj je smanjiti broj kombinacija ulaznih vrijednosti koje sistem mora uzeti u obzir kako bi se postigla efikasnost, brzina i tačnost u izvršavanju zadataka.

3. Slijepo pretraživanje - ukratko objasniti!

Slijepo pretraživanje je jedna od tehnika pretrage u oblasti inteligentnih sistema koja se koristi kada nemamo nikakvo znanje o ciljnom prostoru stanja. Ova tehnika se zasniva na generisanju svih mogućih akcija i stanja sistema, a zatim provjerava koja akcija vodi do ciljnog stanja.

Slijepo pretraživanje se obično koristi u jednostavnim problemima pretrage u kojima je cilj pronaći rješenje u skupu svih mogućih stanja sistema. Postoje tri osnovna pristupa slijepog pretraživanja: pretraživanje u dubinu, pretraživanje u širinu i pretraživanje s ograničenjima.

Slijepo pretraživanje može biti vrlo spor i neefikasan u problemima s velikim prostorom stanja ili složenim problemima. Međutim, može biti korisno kao početni pristup u rješavanju problema pretrage prije prelaska na složenije tehnike.

4. Pretraživanje u širinu - ukrako objasniti!

Pretraživanje u širinu je tehnika pretrage u oblasti inteligentnih sistema koja pretražuje sve čvorove na istoj razini prije nego što nastavi na sljedeću razinu. Ova tehnika koristi red (queue) kako bi održala redoslijed čvorova koji se pretražuju.

Algoritam počinje od početnog stanja i provjerava sve moguće akcije koje se mogu primijeniti na to stanje. Zatim se generišu svi mogući sljedeći čvorovi sistema, koji se zatim dodaju u red pretrage. Algoritam zatim prelazi na sljedeći čvor u redu i ponavlja proces generisanja i dodavanja sljedećih čvorova, sve dok se ne pronađe ciljno stanje.

Pretraživanje u širinu je garantovano da će pronaći najkraći put do ciljnog stanja, ako postoji, ali može biti vrlo sporo i neefikasno za probleme s velikim prostorom stanja. Također, može biti potrebno puno memorije za održavanje čvorova u redu pretrage.

Uprkos svojim nedostacima, pretraživanje u širinu se često koristi kao osnovna tehnika pretrage u problemima koji imaju mali prostor stanja ili koji zahtijevaju pronalaženje najkraćeg puta do ciljnog stanja.

5. Pretraživanje u širinu - ukrako objasniti!

Pretraživanje u dubinu je tehnika pretrage u oblasti inteligentnih sistema koja pretražuje stablo pretrage u dubinu, prije nego što se vrati i pretražuje se drugi dio stabla pretrage. Ova tehnika koristi stog (stack) kako bi održala redoslijed čvorova koji se pretražuju.

Algoritam počinje od početnog stanja i primjenjuje jednu akciju kako bi generisao sljedeće stanje sistema. Tada se pretražuju sljedeći čvorovi sistema povezani s tim stanjem, a zatim se generiše sljedeće stanje sistema i ponavlja se postupak pretrage u dubinu. Kada se dostigne kraj grane stabla pretrage, algoritam se vraća na prethodni čvor na stogu i ponavlja proces pretrage u dubinu za drugu granu stabla pretrage.

Pretraživanje u dubinu može biti brže od pretraživanja u širinu u slučajevima kada se ciljno stanje nalazi u velikoj dubini stabla pretrage, ali algoritam nije garantovan da će pronaći najkraći put do ciljnog stanja. Također, ako postoji beskonačna petlja u stablu pretrage, algoritam može zaglaviti u petlji i nikada ne pronaći ciljno stanje.

Pretraživanje u dubinu se često koristi u problemima koji imaju beskonačno stablo pretrage, poput problema igre sa potezima gdje postoji beskonačan broj mogućih poteza. Međutim, zbog nedostataka ove tehnike pretrage, obično se kombinuje s drugim tehnikama pretrage u složenijim problemima.

6. Informirano pretraživanje - ukratko objasniti!

Informirano pretraživanje je tehnika pretrage u oblasti inteligentnih sistema koja koristi dodatne informacije o problemu kako bi vodila pretragu prema ciljnom stanju. Ove dodatne informacije se obično nazivaju heuristike i koriste se kako bi se procijenilo koliko je dobro neko stanje sistema u smislu približavanja ciljnom stanju.

Najčešće korištena heuristika u informiranom pretraživanju je heuristika procjene razdaljine do ciljnog stanja. Ova heuristika procjenjuje koliko je udaljeno neko stanje sistema od ciljnog stanja. Algoritam koristi ovu heuristiku kako bi prilagodio proces pretrage i pretražuje stanja koja su bliža ciljnom stanju prije nego što pretražuje stanja koja su dalja.

Najpoznatiji algoritam informiranog pretraživanja je A* algoritam. Ovaj algoritam koristi heuristiku procjene razdaljine do ciljnog stanja i koristi je kako bi prilagodio proces pretrage. A* algoritam kombinuje informacije o troškovima do trenutnog stanja sistema i heurističku procjenu preostalog troška do ciljnog stanja kako bi donio odluku o sljedećem koraku u pretrazi.

Informirano pretraživanje obično je efikasnije od slijepog pretraživanja, posebno u problemima s velikim prostorom stanja, jer koristi dodatne informacije o problemu kako bi se smanjio broj generisanih stanja sistema.

7. Zadatak: Riješiti problem putovanja na području Bosne i Hercegovine korištenjem grafa stanja (dio mape će biti zadan sa oznakama udaljenosti)

Problem putovanja na području Bosne i Hercegovine može se riješiti korištenjem grafa stanja. Graf se sastoji od čvorova koji predstavljaju gradove u Bosni i Hercegovini i bridova koji predstavljaju ceste koje spajaju ove gradove. Svaki brid ima oznaku udaljenosti koja predstavlja udaljenost između dvije povezane tačke na mapi.

Za rješavanje ovog problema, može se primijeniti algoritam pretrage grafa stanja, poput algoritma pretrage u širinu ili algoritma pretrage u dubinu, ali bi bilo efikasnije primijeniti informirano pretraživanje kao što je A* algoritam, uz korištenje heuristike koja procjenjuje udaljenost od trenutnog grada do ciljnog grada.

Za primjer, možemo postaviti cilj da se putuje iz Sarajeva do Banja Luke. A* algoritam bi započeo u Sarajevu i generisao sve moguće ceste koje vode iz Sarajeva do drugih gradova. Za svaku novu tačku u grafu stanja, algoritam bi izračunao heurističku procjenu preostale udaljenosti do ciljnog grada, u ovom slučaju Banja Luke. Algoritam bi koristio ove procjene da bi odlučio koji put prema drugim gradovima ima najmanju cijenu. Postupak bi se ponavljao sve dok se ne bi stiglo do ciljnog grada.

Na primjer, pretpostavimo da postoji cesta iz Sarajeva do Tuzle, sa oznakom udaljenosti 120 kilometara, cesta iz Sarajeva do Mostara, sa oznakom udaljenosti 150 kilometara, i cesta iz Tuzle do Banja Luke, sa oznakom udaljenosti 100 kilometara. Heuristika bi procijenila da je udaljenost između Tuzle i Banja Luke 150 kilometara, pa bi A* algoritam odabrao cestu od Sarajeva do Tuzle, a zatim cestu od Tuzle do Banja Luke, jer bi ukupna udaljenost bila manja nego da se krenulo cestom od Sarajeva do Mostara.

Ovaj pristup omogućava efikasnije rješavanje problema putovanja u Bosni i Hercegovini, jer omogućava algoritmu da pretražuje samo one ceste koje imaju najmanju cijenu, umjesto da pretražuje sve ceste u Bosni i Hercegovini.

8. Šta je to naivni Bayes-ov klasifikator unutar Inteligentnih sistema - ukratko objasniti!

Naivni Bayes-ov klasifikator je algoritam mašinskog učenja koji se koristi za klasifikaciju objekata u različite klase na osnovu atributa koji su im pridruženi. Ovaj klasifikator koristi Bayesovu teoremu kako bi izračunao vjerovatnoću da objekat pripada određenoj klasi na osnovu poznatih vrijednosti atributa. "Naivni" u nazivu ovog klasifikatora znači da se pretpostavlja da su svi atributi nezavisni jedni od drugih.

Bayesov teorem opisuje kako se vjerovatnoća da se neki događaj A dogodi, mijenja kada se uzme u obzir neki drugi događaj B. U kontekstu klasifikacije, Bayesov teorem se koristi kako bi se izračunala vjerovatnoća da objekat pripada određenoj klasi, kada su poznati atributi objekta.

Naivni Bayes-ov klasifikator radi tako što prvo nauči vjerovatnoću svakog atributa za svaku klasu, iz podataka koji su mu dati. Zatim, kada se na osnovu novog skupa atributa dobije objekat za klasifikaciju, klasifikator koristi Bayesov teorem da bi izračunao vjerovatnoću da objekat pripada određenoj klasi. Klasa sa najvećom vjerovatnoćom se zatim dodjeljuje objektu.

Naivni Bayes-ov klasifikator se često koristi za tekstualnu klasifikaciju, kao što je klasifikacija email poruka u spam i ne-spam. U ovom slučaju, atributi su riječi u email poruci, a klase su "spam" i "ne-spam". Klasifikator se može trenirati na velikom skupu email poruka, kako bi naučio vjerovatnoću svake riječi za svaku klasu. Zatim, kada se dobije nova email poruka, klasifikator koristi naučene vjerovatnoće da bi izračunao vjerovatnoću da je email poruka "spam" ili "ne-spam".
Naivni Bayes-ov klasifikator je jednostavan za implementaciju i ima dobru performansu na velikom broju problema klasifikacije. Međutim, zbog pretpostavke da su svi atributi nezavisni, ponekad može biti manje tačan u situacijama kada postoji zavisnost među atributima.

9. Šta su to klasteri i koje metode/tehnike se najčešće koriste za analizu klastera?

Klasteri su grupisanja sličnih objekata u skupove (klasteri) na osnovu zajedničkih karakteristika. U oblasti inteligentnih sistema, klasteri se često koriste za analizu podataka, otkrivanje skrivenih obrazaca i segmentaciju tržišta. Na primjer, može se koristiti klasterska analiza za grupisanje korisnika na osnovu zajedničkih karakteristika, kao što su godine, obrazovanje, prihod, itd.

Postoji nekoliko tehnika za analizu klastera, a neke od najčešćih su:

1. K-means klasterovanje: Ovo je jedna od najčešćih tehnika za analizu klastera. Ova metoda radi tako što se prvo izabere broj klastera koji se žele pronaći. Zatim se nasumično odabiru tačke u skupu podataka kao početne tačke za svaki klaster. Svaka tačka se zatim dodjeljuje u najbliži klaster, a srednja vrijednost svih tačaka u svakom klasteru se izračunava kako bi se dobila nova početna tačka. Ovaj proces se ponavlja sve dok se centri klastera ne stabiliziraju.

2. Hijerarhijsko klasterovanje: Ova metoda grupiše objekte u hijerarhijsku strukturu, u kojoj se svaki klaster može sastojati od pojedinačnih objekata ili drugih klastera. Postoje dvije vrste hijerarhijskog klasterovanja: aglomerativno i dijeljenje. Aglomerativno klasterovanje počinje sa svakim objektom kao odvojenim klasterom i spaja ih u veće klastera dok ne dobijemo jedan veliki klaster. Dijeljenje klasterovanja radi suprotno - počinje sa jednim velikim klasterom i dijeli ga na manje dok se ne dobiju pojedinačni objekti.

3. Gustinsko klasterovanje: Ova metoda grupiše objekte u klaster na osnovu sličnosti u gustoći podataka. Ova metoda se najčešće koristi za analizu prostornih podataka, kao što su slike ili geografski podaci.

4. Model-based klasterovanje: Ova metoda koristi statističke modele da bi se pronašli klasteri u podacima. Model-based klasterovanje može biti korisno u situacijama kada su podaci izuzetno kompleksni i kada se ne mogu jasno grupisati pomoću drugih metoda.

Kada se odabere metoda za analizu klastera, ključno je izabrati odgovarajući broj klastera za analizu. To se obično radi pomoću metoda kao što su "elbow method" ili "silhouette method", koje pomažu u određivanju optimalnog broja klastera. Elbow method koristi metriku SSE (suma kvadrata udaljenosti) kako bi se pronašao broj klastera nakon kojeg se smanjenje udaljenosti između podataka unutar klastera više ne smanjuje značajno. Silhouette method takođe koristi metriku udaljenosti između podataka, ali pravi grafikon silhouette koeficijenta, koji se kreće od -1 do 1 i pokazuje koliko su podaci u jednom klasteru slični jedni drugima, u poređenju sa podacima u drugim klasterima. Optimalni broj klastera je onaj za koji je srednji silhouette koeficijent najveći.

10. Šta su to stabla odluke (Decision Tree) - ukratko objasniti!

Stabla odluke su grafički prikazani modeli odlučivanja u kojima se koriste stabla radi predviđanja klasifikacije ili vrijednosti ciljne varijable na osnovu skupa ulaznih podataka. U ovim stablima

svaki čvor predstavlja atribut, a grane predstavljaju vrijednosti tog atributa. Koren čvor predstavlja početnu varijablu, dok se listovi stabla koriste za predviđanje vrijednosti ciljne varijable.

Postoje različite metode za izgradnju stabala odluke, a neke od najpopularnijih uključuju ID3 (Iterative Dichotomiser 3), C4.5 i CART (Classification and Regression Trees). Ove metode se obično koriste za klasifikaciju podataka u različitim oblastima, uključujući poslovne, medicinske i finansijske sektore. Stabla odluke su popularni zbog svoje jednostavnosti i lakoće interpretacije, što ih čini korisnim alatom za donošenje odluka u različitim situacijama.

11. Zadatak: Za problem - regresija za zadani skup podataka:

X: -1.84 -1.52 -1.49 -1.18 -0.66 -0.15 -0.02 0.42 0.92 1.34 1.57

Y: 3.00 2.60 1.50 0.50 1.30 0.65 1.70 -0.25 -0.45 -1.00 -2.25

potrebno je pronaći odgovarajući model y = f (x) kako bi iz modela mogli dobiti informaciju
koliki će biti y za x-eve koji nisu u skupu primjeraka za učenje

Za pronalaženje odgovarajućeg modela regresije potrebno je najprije vizualizirati podatke pomoću grafa. Na ovaj način možemo vizualno procijeniti kakva je relacija između x i y varijable.

Na temelju ovog grafa, možemo zaključiti da postoji negativna korelacija između x i y varijable. To znači da kada se x povećava, y se smanjuje. Kada smo utvrdili ovu relaciju, možemo odabrati model regresije koji najbolje opisuje ove podatke.

Za pronalaženje odgovarajuće funkcije regresije za ovaj skup podataka možemo koristiti algoritme strojnog učenja kao što je linearna regresija ili polinomijalna regresija.

Koristeći algoritam linearne regresije, možemo modelirati funkciju y = f(x) kao:

y = w0 + w1*x *gdje su w0 i w1 parametri koje treba naučiti iz skupa podataka.*

Cilj linearne regresije je pronaći vrijednosti parametara w0 i w1 tako da minimiziramo sumu kvadrata greške između predviđenih vrijednosti y i stvarnih vrijednosti y u skupu podataka za učenje.

Konkretno, to se može postići primjenom metode najmanjih kvadrata, koja minimizira sumu kvadrata greške, odnosno sumu kvadrata razlika između predviđenih i stvarnih vrijednosti:

min sum((y_pred - y)^2)

a) *Označite x-osi i y-osi grafikona.*
b) *Na x-osi označite vrijednosti -2, -1.5, -1, -0.5, 0, 0.5, 1, 1.5 i 2.*
c) *Na y-osi označite vrijednosti -3, -2, -1, 0, 1, 2 i 3.*
d) *Označite točke iz skupa podataka na grafikonu.*

e) Nacrtajte liniju koja prolazi kroz točke.

I trebala bi se dobiti funkcija ***y = 0.524 - 1.218*x.***

12. Šta su to neuronske mreže?

Neuronske mreže su skupovi algoritama strojnog učenja koji su inspirirani strukturom mozga sisavaca. One se sastoje od skupova povezanih jedinica (neurona) koji obrađuju i prenose informacije putem svojih međusobnih veza. Svaki neuron prima ulazne signale, obrađuje ih i zatim šalje izlazne signale drugim neuronima u mreži. Na taj način, neuronske mreže su sposobne naučiti i prepoznati složene obrasce u podacima, poput slika, zvuka ili teksta.

Neuronske mreže se sastoje od slojeva neurona, gdje svaki neuron prima ulazne podatke, obrađuje ih pomoću neke aktivacijske funkcije i prosljeđuje ih drugim neuronima u mreži. Postoji više vrsta slojeva, poput slojeva za ulazne podatke, skrivenih slojeva i slojeva izlaznih podataka. Uz to, neuronske mreže se uče kako bi se prilagodile novim ulaznim podacima kroz postupak učenja, poput algoritama unazadne propagacije greške.

13. Vještačka neuornska mreža i Van Neumannova arhitektura – poređenje?

Van Neumannova arhitektura i neuronske mreže su dva različita koncepta u oblasti računarstva. Van Neumannova arhitektura se odnosi na tradicionalan računarski sistem koji koristi centralnu obradu (CPU) za izvršavanje algoritama i upravljanje memorijom. To je arhitektura koja se koristi u većini računara današnjice.

S druge strane, neuronske mreže su vrsta paralelnog računarstva inspirirana biološkim neuronima. Neuronske mreže se sastoje od velikog broja procesnih jedinica (neurona) koji rade zajedno za rješavanje problema. Svaki neuron prima signale od drugih neurona, obrađuje ih i zatim šalje signale drugim neuronima. Neuronske mreže su projektirane za obavljanje različitih zadataka, kao što su klasifikacija, predviđanje i obrada slike.

Ključna razlika između Van Neumannove arhitekture i neuronskih mreža je u tome što neuronske mreže imaju paralelnu arhitekturu, što znači da svi neuroni mogu raditi istovremeno i obrađivati veliku količinu podataka u kratkom vremenskom razdoblju. To čini neuronske mreže vrlo brzim i efikasnim za obradu podataka koji se koriste u velikom broju aplikacija, uključujući prepoznavanje uzoraka, prevođenje jezika, klasifikaciju slika, preporučivanje proizvoda i drugo. Uz to, neuronske mreže su sposobne učiti i prilagođavati se novim situacijama, što ih čini izuzetno fleksibilnim i korisnim u situacijama kada se suočavamo sa složenim i dinamičkim problemima. Nasuprot tome, Van Neumannova arhitektura koristi serijsku obradu podataka, što znači da podaci prolaze kroz procesor jedan po jedan, čime se smanjuje brzina i efikasnost obrade podataka.

14. Primjena vještačkih neuronskih mreža - navesti nekoliko primjera!

Vještačke neuronske mreže (VNM) imaju široku primjenu u mnogim područjima i industriji. Neke od primjena su:

1. Prepoznavanje uzoraka: VNM se često koriste za prepoznavanje uzoraka u slikama, zvuku i tekstu. Primjeri uključuju prepoznavanje rukom pisanih slova i brojeva, prepoznavanje lica i prepoznavanje govora.

2. Financije: VNM se koriste u finansijskom sektoru za predviđanje cijena dionica, kretanja tržišta i otkrivanje prijevara.

3. Medicina: VNM se koriste u medicinskim istraživanjima za analizu genoma, dijagnostiku i predviđanje bolesti. Primjeri uključuju dijagnostiku raka, predviđanje rizika od srčanih bolesti i praćenje zdravlja pacijenata.

4. Proizvodnja: VNM se koriste u proizvodnoj industriji za optimizaciju proizvodnih procesa, praćenje kvalitete i predviđanje kvarova strojeva.

5. Sigurnost: VNM se koriste u različitim sigurnosnim aplikacijama, kao što su prepoznavanje lica, praćenje kretanja i otkrivanje prijevara u bankovnim transakcijama.

6. Transport: VNM se koriste u industriji transporta za optimizaciju ruta, predviđanje kašnjenja i praćenje vozila.

Ovo su samo neki primjeri primjene VNM-a, a ova tehnologija se sve više koristi u mnogim drugim područjima kao što su sport, umjetna inteligencija, robotika, marketing i druge industrije.

15. TLU perceptron - skicirajte model i ukratko objasnite!

TLU (Threshold Logic Unit) perceptron je jednostavan model vještačkog neurona koji se koristi za binarnu klasifikaciju. Sastoji se od više ulaza i jednog izlaza. Svaki ulaz je povezan s težinom koja može biti pozitivna ili negativna. Izlaz neurona se izračunava kao suma svih ulaza pomnoženih s njihovim težinama, te se zatim primjenjuje prag ili određena funkcija aktivacije. Ako je izlaz veći od praga, neuron izbacuje izlaznu vrijednost 1, inače 0.

Matematički, TLU perceptron možemo opisati izrazom:

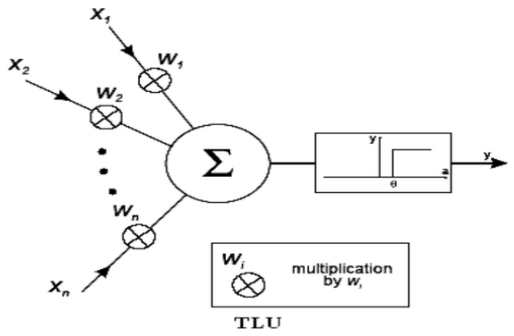
y = 1, ako (w1*x1 + w2*x2 + ... + wn*xn) > theta

y = 0, inače

gdje su w1, w2, ..., wn težine ulaza, x1, x2, ..., xn ulazi, a theta je prag.

Model TLU perceptrona se često koristi u problemima binarne klasifikacije, kao što su prepoznavanje oblika, klasifikacija slika i analiza teksta.

The Threshold Logic Unit (TLU), McCulloch&Pitts, 1943 is the simplest model of an artificial neuron.



16. Zadatak: ***Korištenjem algoritma perceptrona riješiti sljedeći zadatak. Skup uzoraka za učenje klasifikacijskog problema. Za uzorke koji pripadaju klasi K1 želimo da neuron na izlazu generira vrijednost 1 a za uzorke koji pripadaju klasi K2 želimo da neuron na izlazu generira vrijednost -1.***

Redni broj uzorka Ulaz (x2 , x1) Željeni izlaz t

1. (2, 5) 1

2. (5, 2) 1

3. (1, 5) -1

4. (5, 1) -1

Da bismo riješili ovaj klasifikacijski problem koristimo TLU perceptron, gdje će funkcija prijenosa biti definirana kao:

f(net) = 1, ako net >= 0

f(net) = -1, ako net < 0

Gdje je net definiran kao:

net = w1*x1 + w2*x2

Algoritam perceptrona će se koristiti za određivanje vrijednosti parametara w1 i w2 tako da minimiziramo pogrešku klasifikacije.

Početne vrijednosti parametara možemo postaviti na nulu, tj. w1 = 0 i w2 = 0.

Prvo ćemo obraditi uzorak 1. Uzorak 1 pripada klasi K1, tako da je željeni izlaz t1 = 1. Trenutni izlaz neurona za ovaj uzorak možemo izračunati na sljedeći način:

net1 = w1*x1 + w2*x2 = 0*2 + 0*5 = 0

y1 = f(net1) = f(0) = 1

Kako je y1 različito od t1, pogreška klasifikacije je:

e1 = t1 - y1 = 1 - 1 = 0

Sada možemo izračunati nove vrijednosti parametara w1 i w2 koristeći formulu:

wi = wi + η*e*x(i)

Gdje je η stopa učenja (learning rate) koja obično ima vrijednost između 0 i 1.

Možemo postaviti η = 1 za ovaj primjer.

Stoga će se nove vrijednosti parametara w1 i w2 izračunati na sljedeći način:

w1 = w1 + η*e1*x1 = 0 + 1*0*2 = 0

w2 = w2 + η*e1*x2 = 0 + 1*0*5 = 0

Nakon prvog koraka algoritma perceptrona, parametri su ostali nepromijenjeni, što znači da nema potrebe za daljnjim učenjem.

Kada se primijeni ovaj model na nove ulazne vrijednosti, ako je izlaz veći od ili jednak 0, tada uzorak pripada klasi K1, a ako je manji od 0, tada pripada klasi K2.

17. Objasnite algoritam kod perceptron neuronske mreže!

Kod perceptron neuronske mreže je algoritam za učenje nadzirano učenje. Cilj algoritma je naučiti perceptron kako klasificirati primjere iz skupa podataka na temelju njihovih značajki.

Algoritam kod perceptrona se sastoji od sljedećih koraka:

1. Inicijaliziraj težine w i pristranost b na slučajne vrijednosti.

2. Odaberi primjer iz skupa za učenje i izračunaj izlaz perceptrona pomoću aktivacijske funkcije, koja ovisi o težinama i pristranosti.

3. Usporedi izlaz perceptrona s željenim izlazom za taj primjer i izračunaj grešku.

4. Prilagodi težine i pristranost na temelju izračunate greške i vjerojatnosti pripadnosti primjera različitim klasama.

5. Ponovi korake 2-4 za sve primjere u skupu za učenje, više puta ako je potrebno, dok se ne postigne željena razina točnosti klasifikacije.

Cilj algoritma je minimizirati grešku klasifikacije na skupu za učenje kako bi se dobila što bolja općenitost perceptrona, tj. da što točnije klasificira nove, neviđene primjere izvan skupa za učenje. Algoritam kod perceptrona je jedan od osnovnih algoritama za učenje neuronskih mreža i može se primijeniti u mnogim problemima klasifikacije

18. Objasnite algoritam backprobagation (algoritam propagacije greške unazad).

Backpropagation algoritam je jedan od najvažnijih algoritama za učenje neuronskih mreža, posebno mreža s više slojeva. Ideja je da se prvo mreža slučajno inicijalizira, a zatim se koristi skup za učenje kako bi se procijenila greška između stvarnog i predviđenog izlaza.

Algoritam backpropagation se sastoji od nekoliko koraka:

1. Inicijalizacija težina: Težine neuronskih veza se slučajno inicijaliziraju.

2. Prikupljanje podataka: Skup za učenje se koristi kako bi se prikupili ulazni i željeni izlazni podaci.

3. Prolaz naprijed (forward pass): Ulazni podaci se prosljeđuju kroz neuronsku mrežu i dobivaju se predviđeni izlazi.

4. Izračunavanje greške: Razlika između stvarnog i predviđenog izlaza se računa kako bi se procijenila greška.

5. Prolaz unazad (backward pass): Greška se propagira unazad kroz neuronsku mrežu kako bi se izračunale izvedene vrijednosti težina.

6. Ažuriranje težina: Težine se ažuriraju kako bi se minimizirala greška između stvarnog i predviđenog izlaza.

7. Ponavljanje procesa: Koraci 3-6 se ponavljaju sve dok se greška ne smanji dovoljno ili dok se ne postigne maksimalan broj iteracija.

Cilj backpropagation algoritma je minimizirati grešku između stvarnog i predviđenog izlaza neuronske mreže. Ovaj algoritam je vrlo učinkovit za učenje mreža s više slojeva, jer omogućava optimizaciju težina na svim slojevima, a ne samo na izlaznom sloju.

19. Kako se neuronske mreže mogu podijeliti?

Neuronske mreže se mogu podijeliti na više načina, a neke od najčešćih podjela su:

1. Prema broju slojeva: mogu biti jednoslojne (engl. single-layer) ili višeslojne (engl. multi-layer) mreže. Jednoslojne mreže sastoje se samo od ulaznog sloja i izlaznog sloja, dok višeslojne mreže imaju najmanje jedan skriveni sloj između ulaznog i izlaznog sloja.

2. Prema načinu povezivanja neurona: mogu biti potpuno povezane (engl. fully connected) ili rijetko povezane (engl. sparsely connected) mreže. Potpuno povezane mreže imaju svaki neuron povezan sa svakim neuron u susjednom sloju, dok rijetko povezane mreže imaju samo neke veze između neurona.

3. Prema vrsti zadatka: neuronske mreže se mogu koristiti za rješavanje različitih vrsta zadataka, kao što su klasifikacija, regresija, segmentacija slika, generiranje slika, prepoznavanje govora itd.

4. Prema aktivacijskoj funkciji: različite vrste aktivacijskih funkcija se koriste u neuronskim mrežama, kao što su sigmoidna funkcija, tangens hiperbolni, ReLU, softmax itd. Ovisno o problemu koji se rješava, odabire se odgovarajuća aktivacijska funkcija.

20. Navesti bar po dva primjera statičkih, dinamičkih i neizrazitih neuronskih mreža i ukratko objasniti!

Statičke neuronske mreže:

1. Perceptron - jednostavan model binarne klasifikacije koji se sastoji od samo jednog neurona.

2. Multi-layer perceptron (MLP) - višeslojna neuronska mreža sastavljena od jednog ili više skrivenih slojeva, često korištena za rješavanje problema klasifikacije i regresije.

Dinamičke neuronske mreže:

1. Hopfieldova neuronska mreža - jedna od najjednostavnijih dinamičkih neuronskih mreža koja se koristi za memoriranje i prepoznavanje uzoraka.

2. Rekurentna neuronska mreža (RNN) - neuronska mreža koja se sastoji od petlji unutar neurona, čime omogućuje obradu sekvenci podataka, poput prirodnog jezika ili vremenskih serija.

Neizrazite neuronske mreže:

1. Kohonenova mreža - samo-organizirajuća sekvencijalna mreža koja koristi neizrazite metode za klasifikaciju i grupiranje podataka.

2. Mreže sa zadovoljstvom ograničenom energijom (RBMs) - neizrazite neuronske mreže koje se koriste za učenje skrivenih reprezentacija podataka.

21. Šta su to jednoslojne, a šta višeslojene neuronske mreže - navesti primjer i ukratko objasniti!

Jednoslojne neuronske mreže su neuronske mreže sastavljene od samo jednog sloja neurona koji prima ulazne vrijednosti, obrađuje ih i generira izlaz. Ovaj sloj naziva se sloj izlaza (output layer) i obično ima jedan ili više neurona. Jednoslojne neuronske mreže najčešće se koriste za jednostavne probleme poput binarne klasifikacije.

Primjer jednoslojne neuronske mreže može biti perceptron, koji je binarni klasifikator koji može klasificirati ulazne podatke u dvije kategorije. Sastoji se od samo jednog sloja neurona koji prima ulazne vrijednosti i izlazni sloj koji daje izlaznu vrijednost 0 ili 1, ovisno o tome kojoj kategoriji pripada ulazni podatak.

Višeslojne neuronske mreže su neuronske mreže sastavljene od više slojeva neurona koji obrađuju ulazne vrijednosti i generiraju izlaz. Ove mreže se obično koriste za složenije probleme poput prepoznavanja objekata, prevođenja jezika i preporučivanja sadržaja.

Primjer višeslojne neuronske mreže je višeslojna perceptronska mreža (Multi-Layer Perceptron - MLP), koja ima najmanje tri sloja neurona - sloj ulaza (input layer), jedan ili više skrivenih slojeva (hidden layer) i sloj izlaza (output layer). MLP se obično koristi za klasifikaciju ili regresiju.

22. Kako se rad neuronskih mreža može prikazati - koje su faze rada i karakteristike sve od faza?