Univerza v Ljubljani

UNIVERZA V LJUBLJANI
NARAVOSLOVNOTEHNIŠKA FAKULTETA
ODDELEK ZA TEKSTILSTVO, GRAFIKO IN OBLIKOVANJE

STROJNO UČENJE S POMOČJO JAVASCRIPT PROGRAMSKEGA JEZIKA

AVTOR: ŽIGA KRAŠOVEC, GMT-3
MENTOR: PRED. MIHAEL LAZAR

JESENICE, 25. 04. 2018

Kazalo vsebine

1 Uvod

Umetna inteligenca (artificial intelligence - AI) je vse bolj in bolj popularna tematika v svetu informacijskih tehnologij, saj nam omogoča reševanje problemov iz drugačne, bolj organske perspektive in posledično reševanje problemov, ki jih pred tem ni bilo mogoče reševati, vsaj ne na nek smiselen način. V kontektsu umetne inteligence pogosto zasledimo še izraza strojno učenje (machine learning - ML) in globoko učenje (deep learning - DL). Kaj točno vsak od teh izrazov predstavlja bom med drugim opisal v tej seminarski nalogi. Strojno učenje je predvsem učinkovito na področju analize velike količine podatkov (big data analysis), kjer so podatki, ki v sistem vstopajo (inputs) lahko zelo različni glede na problem, ki ga rešujemo. Torej, razlika med klasičnimi programerskimi algoritmi in algoritmi, ki jih vključuje strojno učenje je predvsem v tem, da se pri strojnem učenje nemesto striktnih ukazov zanašamo na organsko "učenje" sistema, ki ga dosežemo s pomočjo postopnega prilagajanja sistema samega glede na rezultat, ki ga dobimo in primerjamo z znanimi rezultati. Pomembno je izpostaviti, da umetna inteligenca sama po sebi (vsaj zaenkrat) še ni dejansko inteligentna; računalnik kot tak je še vedno "neumen". Njegova prednost leži v hitrosti s katero je sposoben procesirati podatke in le to izkoriščamo pri strojnem učenju, kjer v sistem vnesemo veliko število znanih izidov in tako sistem "naučimo" kaj je prav in kaj je narobe. Izraz umetna inteligenca se med drugim uporablja tudi zato, ker z algoritmi, prisotnimi v procesu strojnega učenja, posnemamo delovanje človeških možganov. To naredimo prek t.i. nevronskih mrež (neural networks), katerih najmanjša enota je nevron (perceptron), ki je odgovoren za obdelavo določenega podatka ali seta podatkov.

Ker je veliko prej omenjenih pojmov precej abstraktnih, bom v seminarski nalogi delovanja takega sistema predstavil prek preprostega sistema določanja optimalne barve teksta (črna ali bela) glede na barvo ozadja (RGB). Sistem sam po sebi je zelo preprost in niti ne bi potreboval nevronske mreže za sprejemljiv rezultat, a se prek njega vseeno na preprost in razumljiv način da prikazati delovanje strojnega učenja s pomočjo nevronskih mrež. Kot programski jezik bom uporabil JavaScript, ki omogoča ogled rezultatov v brskalniku in ob enem interakcijo uporabnika neposredno s sistemom samim.

2 Umetna inteligenca

Področje umetne inteligence ima, za marsikoga presenetljivo, svoje korenine že iz sredine 20. stoletja. Pravzaprav je ta tema aktualna skoraj od nastanka prvih računalnikov dalje, saj je le ta človeku vlil nova področja domišljije, česa vse je tehnologija sposobna. Do razvoja umetne inteligence v pravem pomenu besede pa je prišlo šele pred kratkim, predvsem iz dveh razlogov, in sicer lahka dostopnost velike količine podatkov in pa visoka zmogljivost računalnikov oziroma njihove procesorske zmožnosti. Problemi, ki jih umetna inteligenca rešuje, so predvsem s področja analize, obdelave in (ključno) predvidevanja podatkov. V kontekstu umetne inteligence pogosto zasledimo več različnih izrazov; umetna inteligenca, strojna inteligenca, strojno učenje, globoko učenje, nevronska mreža itd.

2. 1 Pojem umetna inteligenca

Pojem umetna inteligenca vključeje vse stroje (računalnike), ki so sposobni opravljati naloge, ki ponazarjajo karakteristike človeške inteligence. Druga zelo popularna definicija umetne inteligence je, da le ta predstavlja zmožnost učenja brez točno določenih programskih ukazov. Je zelo splošen pojem; vključuje stvari kot so planiranje, razumevanje jezika, prepoznavo objektov in zvokov, učenje in reševanje problemov. V grobem umetno inteligenco delimo na dva glavni panogi; splošna, ki naj bi zavzemala vse karakteristike človeške inteligence in pa ozka, ki vključuje le določene karakteristike in le te opravlja izjemno dobro. Primer ozko usmerjene umetne inteligence bi bil na primer program, ki je zadolžen za prepoznavo slik in ne zna opravljati ničesar drugega, je pa pri tem opravilu zelo učinkovit.

2. 2 Pojem strojno učenje

Strojno učenje oziroma *machine learning (ML)* je preprosto način doseganja umetne inteligence. Strojno učenje predstavlja programske postopke, ki ne zahtevajo točno določenih ukazov za opravljanje določenih opravil, temveč se le teh algoritmov postopoma "naučijo" s pomočjo t.i. *treniranja algoritma*. To treniranje vključuje dovajanje ogromne količine podatkov in dovoljenje algoritmu, da se na podlagi znanih izidov dovedenih podatkov v primerjavi s svojimi rezultati (torej na podlagi odstopanj / napak) prilagaja in izboljšuje čez čas. Dober primer strojnega učenja je drastična izboljšava t.i. "computer vision" - pogleda računalnika (sposobnost računalnika, da prepozna objekt v sliki ali videu); v sistem dovedemo veliko število slik, ki imajo znan rezultat (označene so bile s strani človeka, npr. ali se na sliki pojavlja avto ali ne), algoritem pa nato poizkuša čim bolj natančno določiti model, ki je sposoben brez posega človeka označiti sliko (torej ali se na nej pojavlja avto ali ne) na ravni človeške inteligence (ali pa celo bolje). Ko je ta natačnost enkrat dovolj visoka, rečemo da se je računalnik "naučil" kako izgleda avto na sliki.

2. 3 Pojem globoko učenje

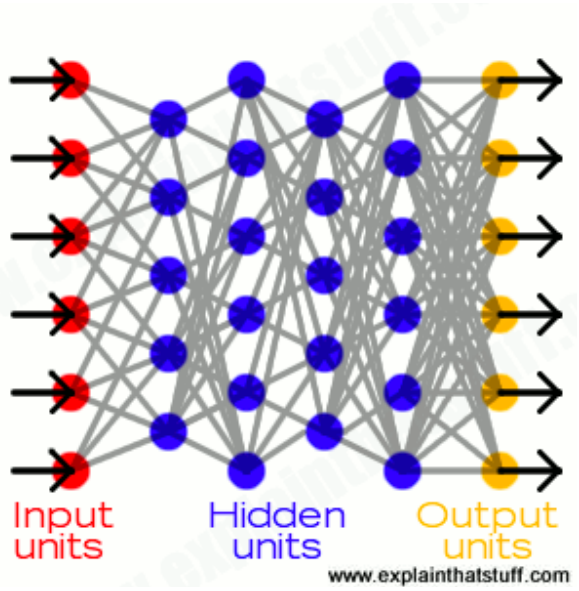
Globoko učenje je le eden od možnih pristopov k strojnemu učenju. Nekateri izmed drugih pristopov so učenje prek "drevesa odločitev" *(decision tree learning)*, *inductive logic programing, clustering* in drugi. Princip globokega učenja je svoj navdih dobil v delovanju človeških možganov, bolj natančno interpovezovanje večjega števila nevronov. Umetne nevronske mreže (večkrat zasledimo zgolj izraz

nevronske mreže) so algoritmi, ki posnemajo delovanje in biološko strukturo človeških možganov. Nevronske mreže vsebujejo t.i. nevrone, ki vsebujejo skrite sloje in povezave z drugimi nevroni v mreži. Vsak sloj v mreži se nauči opravljati točno določeno specifiko celotnega problema (npr. prepoznovo robov objekta na sliki v enem sloju, prepoznava teksture v drugem...) in ta slojni pristop k učenju je prispeval k poimenovanju tega tipa učenja - globoko učenje.

3 Delovanje umetnih nevronskih mrež

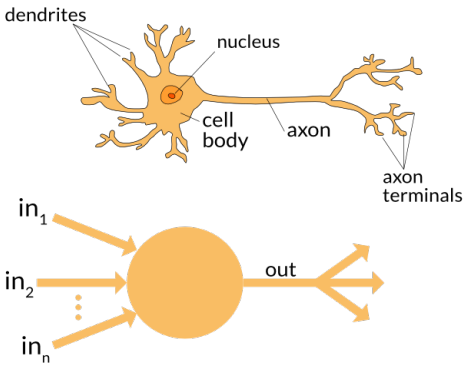
Osnovna ideja umetnih nevronskih mrež je, kot omenjeno, posnemanje delovanja človeških možganov, bolj natančno veliko med seboj povezanih možganskih celic. Za razliko od tradicionalnih algoritmov, nevronske mreže ne morajo biti "sprogramirane " oziroma "nastavljene" na način, ki bi vedno deloval bo predivdeni poti. Tako kot človeški možgani se morajo tudi umetne nevronske mreže "naučiti" kako izvršiti določene zadolžitev, kar se zgodi avtomatsko (v zelo grobem pomenu besede, v ozadju je veliko matematike in algoritmov).

3. 1 Sestava umetne nevronske mreže



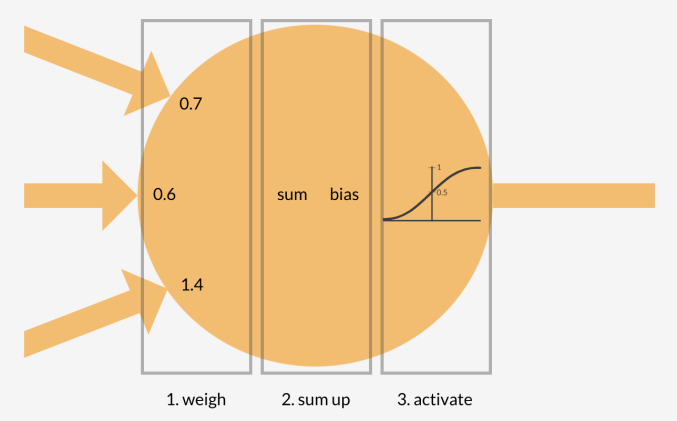
Umetne nevronske mreže so tipično sestavljene vse od nekaj 10 pa vse do več sto, tudi tisoč nevronov, ki so razvrščeni v seriji slojev, vsak sloj pa je povezan tako s prejšnim kot naslednjim slojem. Prvi sloj je zadolžen za dovod različnih podatkov iz "zunanjega sveta" v sistem (inputs), ki jih bo le ta kasneje procesiral in tako ali drugače obdelal. Na nasprotni strani sistema se nahaja zadnji sloj, ki je zadložen za odvod sprocesiranih informacij iz sistema (outputs). Med tema dvema slojema (ki sta obvezna v vsakem sistemu) se nahaja eden ali več t.i. skriti sloj (hidden layer), ki predstavlja jedro "umetnih možganov". Vsi nevroni v teh slojih so med seboj povezani (vsaj v večini sistemov), te povezavo pa označimo s številsko oznako, ki ji rečemo *utež* (weight). Ta številka je lahko ali pozitivna ali negativna (glede na učinek ki ga preda naslednjem sloju v sistemu). Večja kot je številka, večjo pomembnost (efekt) ima enota na naslednjo.

3. 2 Nevron kot enota v umetni nevronski mreži



Najbolj osnovna enota nevronske mreže je *umeten nevron* (v žargonu pogosto zgolj nevron). Tako ime, kot tudi funkcionalnost ki jo nosijo, so dobili po svojih bioloških vzornikih, torej nevronih, ki so prisotni v človeških možganih. In prav tako, kot ima nevron v naših možganih sprejemnike signala, celico, ki te podatke obdela in oddajnike signala, ima tudi umetni nevron nekaj vhodnih kanalov, fazo procesiranja in en izhodni signal, ki ga lahko odda različnemu številu nevronov v naslednjem sloju.

Poglejmo si delovanje enga umetnega nevrona v mreži še bolj podrobno; ključno vpršanje je, kako nevron sprocesira podatke, ki vanj vstopijo. Obdelava teh podatkov je dejansko razmeroma preprosta, in sicer se izvede v treh korakih:



1) **Vsak vhodni signal se ovrednoti**
   Ko signal vstopi v sistem se ovrednoti njegov "efekt", ki ga nosi tako, da se mu določi utež (weight). Vrednost te uteži se tekom učenja sistema nenehno prilagaja gleda na vrednost *napake* sistema.
2) **Vsi signali se seštejejo**
   V tem koraku se signali seštejejo v eno vrednost (vrednosti je dodan tudi delež odstopanja, imenovan *bias*, ki se prav tako prilagaja tekom faze učenja sistema). Tu se ubistvu zgodi vsa "čarovnija". Na začetku so vrednosti uteži in odstopanja povsem naključne, z vsako iteracijo učenja pa se te vrednosti malenkost spremenijo / prilagodijo, da je končni izhodni signal vse bližje željeni vrednosti. Na ta način se sistem postopoma "nauči" željenih vzorcev.
3) **Aktivacija (activation)**
   V zadnjem koraku se rezultat kalkulacije nevrona spremeni v izhodni signal. To dosežemo tako, da rezultat dovedemo v t.i. aktivacijsko funkcijo. Najbolj osnovna oblika aktivacijske funkcije je binarna funkcije, ki lahko vrne le dva možna izida: 0 ali 1. Nevron, čigar aktivacijska funkcija je taka funkcija, se imenuje perceptron in je najbolj osnoven gradnik vsake nevronske mreže.

3. 3 Povezano delovanje nevronov v mreži

Informacije po nevronski mreži potekajo v dve smeri. Informacije so v sistem dovedene prek vstopnih kanalov, ti vspodbudijo skrite sloje ki obdelane informacije predajo izhodnim kanalom. Ta vzorec pogosto imenujemo *feedforward network*. Vsi nevroni se ne "sprožijo" naenkrat, temveč le potem, ko prejmejo posredovane informacije od sloja na levi (prejšnega sloja). Da pa se sistem iz te obdelave lahko nekaj "nauči" mora biti prisoten tudi nek odziv po tem ko sistem enkrat od sebe odda neko informacijo. Primer ponovno lahko prenesemo na človeka; z žogo želimo zadeti koš - po prvem metu si zapomnimo, kako je žoga letela po zraku, s kakšno močjo smo jo vrgli kako visoko smo jo vrgli; met ovrednotimo. Ko vržemo naslednjič se spomnimo prejšnjega meta in ta met nekoliko prilagodimo na podlagi tega, kar smo v prejšnjem metu naredili narobe. Torej smo uporabili naš odziv (feedback) na prejšni met (npr. opazili smo, da smo žogo vrgli prešibko) in na podlagi tega podatko smo prilagodili naš naslednji met (npr. žogo smo vrgli močneje). Večja kot je razlika med našim dejanskim metom in željenim metom, bolj moramo prilagoditi naslednji met. Nevronske mreže se "učijo" na popolnoma enak način, najpogostoje s procesom imenovanim *backpropagation*. Proces vključuje primerjavo rezultata, ki ga je nevronska mreža proizvedla z željenim rezultatom in razliko med tema dvema vrednostima uporabi za prilagajnje uteži sistema, od sloja najbližjega izhodnemu pa vse nazaj do prvega sloja (od tu ime ***back****propagation*). Čez to sistemu omogoči, da se "nauči" obdelati podatke na način, ki proizvede najbolj optimalen rezultat glede na željenega.

4 Primer - napovedovanje barve teksta glede na ozadje

Kot primer bom v seminarski nalogi predstavil napovedovanje barve teksta (črna ali bela) glede na barvo ozadja (r, g, b). Primer je preprost in verjetno niti ne bi potreboval nevronske mreže, a je ravno zato dober kot demonstracija delovanja takega sistema. Ključni dejavniki v sistemu so:

- v nevronsko mrežo vstopijo trije vhodni signali; za vsako barvo v barvnem načinu RGB po en. Ti signali so vrednosti med 0 - 255, katere normaliziramo na vrednost med 0 in 1 (za lažjo obdelavo podatkov).
- ker bo sistem preprost kot izhodni signal pričakujemo le eno od dveh možnosti; ali naj bo barva teksta bela ali črna
- sistem bo vseboval t.i. nadzorovano učenje, tako da mu moramo priskrbeti nek set podatkov, za katere vemo, da so pravilni. Do teh podatkov lahko pridemo na več načinov; optimalno bi bilo, da bi človek za vsako možno barvo v načinu RGB označil, ali je bolj ustrezen bel ali črn tekst. Ker je to seveda neizvedljivo in ker je primer res enostaven, bomo do teh podatkov enostavno prišli prek funkcije, ki nam vrne povprečje seštevka vseh treh barvnih kanalov; torej minimalna vrednost je 0 (črna) maksimalna vrednost je 765 (bela), sredina je torej 382.5. Če je vsota vseh treh kanalov več kot to naj bo barva teksta črna, saj je barva ozadja svetla, če pa je vsota manjša od te vrenosti pa naj bo barva teksta bela, saj je ozadje temno. Pri tem zanemarimo marsikateri faktor in pravila vizualnega oblikovanja, a za naš primer bo zadostovalo.

4. 1 Nevronska mreža v JavaScript kodi

Nevronski mreži, ki je zapisana kot razred (class) moramo najprej določiti začtne vrednosti, kar v JS naredimo znotraj metode **`constructor()`**. Tu določimo število vhodnih kanalov (v našem primeru 3), število nevronov v skritem sloju (prav tako 3) in število izhodnih kanalov (v našem primeru 2 - črna ali bela). Prav tako nastavimo vrednosti uteži in odstopanja na naključno vrednost - le ta se bo nastavljala tekom učenja.

```
15  class NeuralNetwork {
16    constructor(a, b, c) {
17      if (a instanceof NeuralNetwork) {
18        this.input_nodes = a.input_nodes;
19        this.hidden_nodes = a.hidden_nodes;
20        this.output_nodes = a.output_nodes;
21
22        this.weights_ih = a.weights_ih.copy();
23        this.weights_ho = a.weights_ho.copy();
24
25        this.bias_h = a.bias_h.copy();
26        this.bias_o = a.bias_o.copy();
27      } else {
28        this.input_nodes = a;
29        this.hidden_nodes = b;
30        this.output_nodes = c;
31
32        this.weights_ih = new Matrix(this.hidden_nodes, this.input_nodes);
33        this.weights_ho = new Matrix(this.output_nodes, this.hidden_nodes);
34        this.weights_ih.randomize();
35        this.weights_ho.randomize();
36
37        this.bias_h = new Matrix(this.hidden_nodes, 1);
38        this.bias_o = new Matrix(this.output_nodes, 1);
39        this.bias_h.randomize();
40        this.bias_o.randomize();
41      }
42      this.setLearningRate();
43      this.setActivationFunction();
44    }
```

V naši nevronski mreži sta ključnega pomena predvsem sposobnost učenja in sposobnost napovedi rezultata. Torej moramo razredu dodeliti še ti dve metodi.

```
predict(input_array) {
  // Generating the Hidden Outputs
  let inputs = Matrix.fromArray(input_array);
  let hidden = Matrix.multiply(this.weights_ih, inputs);
  hidden.add(this.bias_h);
  // activation function!
  hidden.map(this.activation_function.func);

  let output = Matrix.multiply(this.weights_ho, hidden);
  output.add(this.bias_o);
  output.map(this.activation_function.func);

  return output.toArray();
}
```

Metoda **`predict()`** kot argument prejme *n* število vhodnih signalov. Preden vstopijo v skriti sloj te signale pomnožimo z vrednostjo uteži in jim dodamo vrednost odstopanja. Kot omenjeno v teoretičnem delu, preden signali zapustijo nevron jih moramo "poslati" še prek aktivacijske funkcije, torej preden vrednosti skritega sloja posredujemo izhodnemu sloju le te vnesemo v aktivacijsko funkcijo. Signale posredujemo izhodnemu sloju. Ponovimo prejšne postopke; možnimo jih z utežmi in dodamo vrednost odstopanja in preden jih "vrnemo" jih vnesemo še v aktivacijsko funkcijo. Izhodni podatki bodo tako posredovani v obliki seznama (array), in sicer bo seznam vseboval dve vrednosti - prva bo verjetnost za črno barvo in druga bo verjetnost za belo. To bi v praksi izgledalo približno tako:

```
let outputs = [0.10234, 0.89766]
```

Torej verjetnost, da je barva teksta črne barve je nekaj več kot 10% in verjetnost da je barva teksta bele barve je skoraj 90%. Barva teksta bo torej bele barve.

Da pa do teh napovedi pridemo (in da so pravilne), moramo nevronsko mrežo najprej "natrenirati". To naredimo z metodo **`train()`**. Metoda prejme dva argumenta - vstopne signale in znane vrednosti. Prvi del je podoben prejšnemu primeru - množimo z utežmi, dodamo odstopanje, pošljemo v aktivacijsko funkcijo. Učenje poteka v drugem delu - najprej izračunamo napako izhodnih signalov v primerjavi z znanimi vrednostmi. Izhodne siganale nato vnesemo v aktivacijsko funkcijo, jih pomnožimo z vrednostjo napake in pomnožimo z vrednostjo "hitrosti učenja". Nato izračunamo razliko med prejšno vrednostjo uteži in na novo določeno vrednostjo, in na novo določeno vrednost uteži in vrednost odstopa določimo sistemu. Ta postopek ponovimo za povezavo med skritim slojem in vhodnim slojem. Tako so po vsaki iteraciji vrednosti uteži in odstopanja vedno bolj optimalno nastavljene in sistem je sposoben oddati vedno bolj relavantne rezultate.

```
train(input_array, target_array) {
  let inputs = Matrix.fromArray(input_array);
  let hidden = Matrix.multiply(this.weights_ih, inputs);
  hidden.add(this.bias_h);
  hidden.map(this.activation_function.func);

  let outputs = Matrix.multiply(this.weights_ho, hidden);
  outputs.add(this.bias_o);
  outputs.map(this.activation_function.func);

  let targets = Matrix.fromArray(target_array);

  // ERROR = TARGETS - OUTPUTS
  let output_errors = Matrix.subtract(targets, outputs);

  let gradients = Matrix.map(outputs, this.activation_function.dfunc);
  gradients.multiply(output_errors);
  gradients.multiply(this.learning_rate);

  // Calculate deltas
  let hidden_T = Matrix.transpose(hidden);
  let weight_ho_deltas = Matrix.multiply(gradients, hidden_T);

  this.weights_ho.add(weight_ho_deltas);
  this.bias_o.add(gradients);

  // Hidden layer errors
  let who_t = Matrix.transpose(this.weights_ho);
  let hidden_errors = Matrix.multiply(who_t, output_errors);

  let hidden_gradient = Matrix.map(hidden, this.activation_function.dfunc);
  hidden_gradient.multiply(hidden_errors);
  hidden_gradient.multiply(this.learning_rate);

  // Calcuate input->hidden deltas
  let inputs_T = Matrix.transpose(inputs);
  let weight_ih_deltas = Matrix.multiply(hidden_gradient, inputs_T);

  this.weights_ih.add(weight_ih_deltas);
  this.bias_h.add(hidden_gradient);
 }
}
```

4. 2 Prikazovanje rezultatov v brskalniku

Nevronsko mrežo imamo pripravljeno in jo lahko uporabimo za napoved barve teksta. Prednost tega, da je nevronska mreža spisana v JavaScript programskem jeziku je predvsem v tem, da lahko za ogled rezultatov uporabimo kar brskalnik.

```
// Number of inputs, number of neurons in hidden layer, number of outputs
brain = new NeuralNetwork(3, 3, 2);
```

Najprej ustvarimo nove "možgane" našega sistema - uporabimo razred, ki smo ga naredili prej, mu določimo 3 signale na vhodnem sloju, 3 nevrone v skritem sloju in 2 možna izhodna signala. Kot rečeno, da nevronsko mrežo lahko treniramo potrebujemo neke znane vrednosti za določeno izid, torej spišemo preprosto funkcijo, ki pove željen rezultat:

```
function trainColor(r, g, b) {
  if (r + g + b > 255 * 3 / 2) {
    return [1, 0];
  } else {
    return [0, 1];
  }
}
```

Če bi vseeno želeli ročno pridobiti te znane podatke, smo spisali tudi funkcijo, ki sledi kliku uporabnika in na ta način ustvari znan rezultat - glede na odločitev uporabnika.

```
function trainWithHumanInput() {
  let targets;
  if (mouseX > width / 2) {
    targets = [0, 1];
  } else {
    targets = [1, 0];
  }
  let inputs = [r / 255, g / 255, b / 255];

  brain.train(inputs, targets);
  log('I captured your input');
}
```

Za učenje uporabimo prej omenjeno funkcijo, ki jo ponovimo na vsak frame-rate, ki ga je brskalnik sposoben zagotoviti (torej med 25 - 60 interacij na sekundo). To naredimo s preprostim ukazom znotraj funkcije draw() (specifika knjižnice p5.js ki je uporabljena za prikaz grafike v brskalniku v mojem primeru):

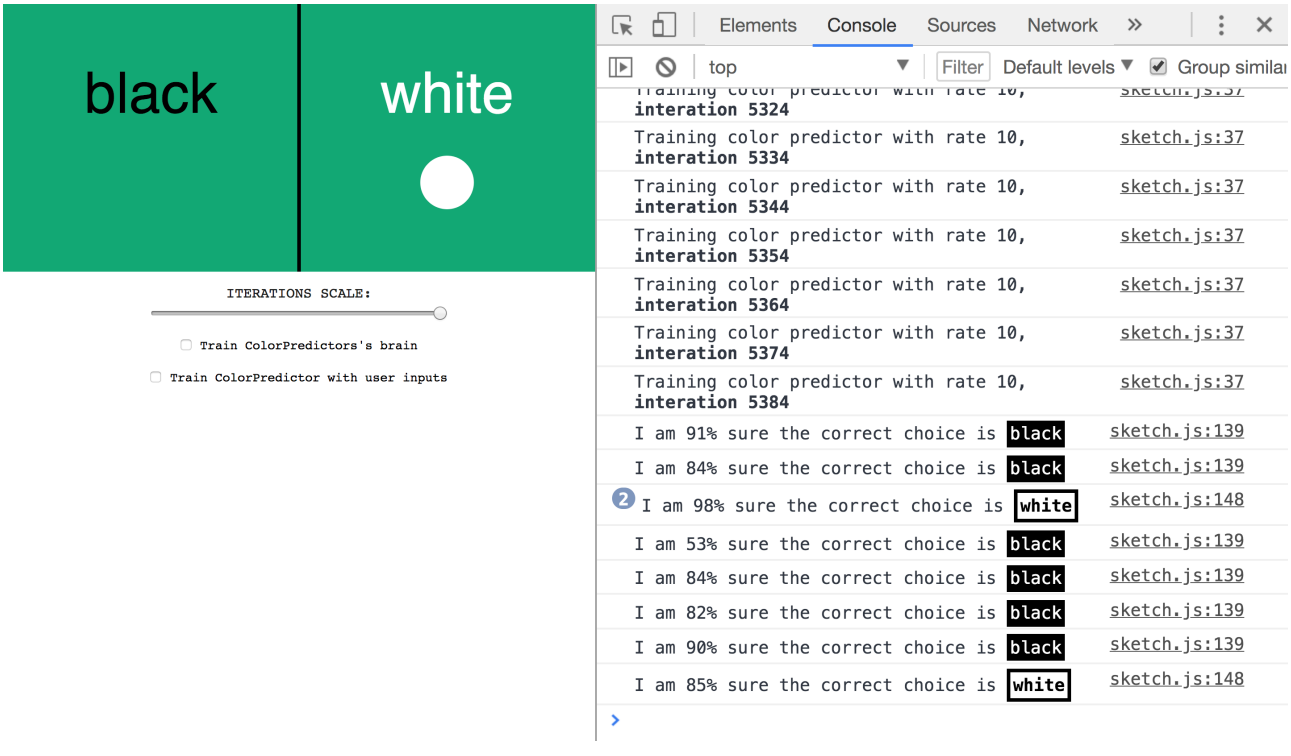
```
if (isTraining.checked()) {
  trainColorPredictor();
} else {
```

Znotraj funkcije trainColorPredictor() zapišemo zanko, v kateri za vsako ponovitev določimo naključno vrednost za vsak barvni kanal med 0 in 255, določimo ciljne vrednosti s pomočjo funkcije, ki smo jo opisli malo prej in sprožimo postopek učenja nevronske mreže, saj imamo na razpolago tako vhodne vrednosti kot znane rezultate. Izgleda tako:

```
for (let i = 0; i < trainingRate.value(); i++) {
  interations++;
  let r = random(255);
  let g = random(255);
  let b = random(255);
  let targets = trainColor(r, g, b);
  let inputs = [r / 255, g / 255, b / 255];
  brain.train(inputs, targets);
}
```

Večkrat ko bomo sprožili postopek učenja nevronske mreže, bolj točne in bolj zanesljive rezultate bomo dobili iz nje.

Za konec si lahko pogledamo še, kako zadeva izgleda v brskalniku:



Kot lahko vidimo, smo nevronsko mrežo "trenirali" z 5384 iteracijami, le ta pa vrača rezultate z razmeroma visoko vrednostjo sigurnosti in pravilnosti (seveda glede na barvo ozadja je bolj ali manj sigurna). Tako smo s preprostim primerom prikazali kompleksen postopek učenja nevronske mreže.

5 Zaključek

Umetna inteligenca in strojno učenje sta vse bolj pogosti obliki obdelave predvsem večje količine podatkov ob enem pa ponujata možnosti za reševanje problem, ki so pred tem veljali za "nerešljive". Z vzponom filozofije in gibanja IoT (internet of things) sta vse bolj in bolj ključnega pomena, saj le ta brez neke inteligence v ozadju ne prinaša nobene dodane vrednosti. Ob enem pa si še vedno velja zapomniti, da za vso to inteligenco in učenjem ne stoji nič drugega, kot premišljena matematika in programska koda, spisana s strani človeka, ki stroj zgolj izkorišča kot orodje za hitrejšo obdelavo podatkov, v našem primeru na zelo premišljen in prefinjen način.

6 Viri

1. Malay Haldar - How Do Neural Networks Work? (povzeto s spletnega mesta: https://medium.com/machine-intelligence-report/how-do-neural-networks-work-57d1ab5337ce, dne 25. 04. 2018)

2. AppliedGo - Perceptrons, the most basic form of a neural network (povzeto s spletnega mesta: https://appliedgo.net/perceptron/, dne 25. 04. 2018)

3. Chris Woodford - Neural networks (povzeto s spletnega mesta: http://www.explainthatstuff.com/introduction-to-neural-networks.html, dne, 25. 04. 2018)

4. Mark Boyd - What Machine Learning Can and Can't Do (povzeto s spletnega mesta: https://thenewstack.io/what-machine-learning-can-and-cant-do/, dne 25. 04. 2018)

5. Calum McClelland - The Difference Between Artificial Intelligence, Machine Learning, and Deep Learning (povzeto s spletnega mesta: https://medium.com/iotforall/the-difference-between-artificial-intelligence-machine-learning-and-deep-learning-3aa67bff5991, dne 25. 04. 2018)

6. Daniel Shiffman - The Nature of Code (povzeto s spletnega mesta: http://wtf.tw/ref/shiffman.pdf, dne 25. 04. 2018)

7. Daniel Shiffman - Neural Networks [serija YouTube posnetkov](povzeto s spletnega mesta: https://www.youtube.com/watch?v=XJ7HLz9VYz0&list=PLRqwX-V7Uu6aCibgK1PTWWu9by6XFdCfh, dne 25. 04. 2018)