Paveikslėlis, kuriame yra juodas, tamsa

Automatiškai sugeneruotas aprašymas

**Kauno technologijos universitetas**

Informatikos fakultetas

**P176B101 Intelektikos pagrindai**

**Laboratorinis darbas Nr. 3**

|  |  |
| --- | --- |
|  |  |
| **Nedas Liaudanskis IFF-1/9**  Studentas | (parašas) (data) |
|  |  |
| **dėst. Nečiūnas Audrius**  Dėstytojas | (parašas) (data) |
|  |  |

**KAUNAS, 2024**

**Turinys**

[1. Įvadas 3](#_Toc165305097)

[ Pirma dalis 3](#_Toc165305098)

[Darbo eiga: 3](#_Toc165305099)

[ Antra dalis 4](#_Toc165305100)

[Darbo eiga: 4](#_Toc165305101)

[2. Duomenų rinkiniai 4](#_Toc165305102)

[ Pirma dalis: 4](#_Toc165305103)

[ Antra dalis: 4](#_Toc165305104)

[3. Funkcinių reikalavimų vykdymas 6](#_Toc165305105)

[ Pirma dalis: 6](#_Toc165305106)

[Duomenų atvaizdavimas 6](#_Toc165305107)

[Tiesinės autoregresijos modelis, tiesinis neuronas 8](#_Toc165305108)

[Modelio verifikacija 9](#_Toc165305109)

[Antrasis scenarijus 15](#_Toc165305110)

[Neuronas 16](#_Toc165305111)

[Papildomi klausymai 17](#_Toc165305112)

[Eksperimentas su skirtingomis modelio eilėmis 17](#_Toc165305113)

[ Kai n = 6 17](#_Toc165305114)

[ Antra dalis: 24](#_Toc165305115)

[Reikšmės prognozavimo modelis 24](#_Toc165305116)

[4. Išvados 27](#_Toc165305117)

1. Įvadas

* Pirma dalis

Šiame laboratoriniame darbe bus pateiktas prognozavimo uždavinio sprendimas, panaudojant tiesinį dirbtinį neuroną. Neuronas bus paremptas tiesine aktyvavimo funkcija (purelin(n)=purelin(Wp+b)=Wp+b). Prognozuojant duomenis bus naudojamas a(k) eilė. Duomenys bus nustatomi pagal eilės ankstines reikšmes a(k-1), a(k-2), ..., a(k-n). Modelį, kurį realizuojame esant prielaidai, kad priklausomybė tarp prognozuojamos reikšmės ir prieš tai esančių eilės elementų gali būti aprašyta naudojant tiesinę funkciją, vadiname autoregresiniu tiesiniu modeliu n-tosios eilės.

Tiesinės autoregresijos modelio išraiška turi šią formą:

Čia , , ∙ ir b yra modelio parametrai ir 𝑎̂(𝑘) pažymi sekančia eilės prognozuojamą reikšmę. Mūsų darbe autoregresinio modelio vaidmenį atliks dirbtinis neuronas, į kuriuo įėjimus padavinėsime prieš tai buvusias eilės reikšmes, modelio parametrų vaidmenį atliks šio neurono svoriniai koeficientai, o prognozuojama reikšmė pasirodys tik neurono išėjime.

čia yra prognozavimo klaida k-tajame laiko žingsnyje. Turint istorinių duomenų rinkinį ieškosime optimalias autoregresinio modelio parametrų reikšmes. Tai reiškia, kad sieksime, kad prognozė, sugeneruota mūsų modelio pagalba, mažiausiai skirtųsi nuo tikrosios 𝑎(𝑘) reikšmės, t.y. kad prognozės klaidų kvadratų suma 𝑒(𝑘) būtų mažiausia visam duomenų rinkiniui. Tuomet sukurtą modelį bus galima naudoti sekančioms reikšmėms prognozuoti.

Prognozavimui naudosime duomenų rinkinį sunspot.txt, kurie pateikia duomenis apie saulės dėmių aktyvumą.

Darbo eiga:

1. Nubrėžti dėmių aktyvumo už 1700- 2014 metus grafiką.
2. Aprašyti matricas P ir T, kuriose atitinkamai pateikiami (mokymosi) įvesties duomenys o taip pat išvesties duomenys.
3. Nubrėžti trimatę diagramą, joje vaizduojant įvesties ir išvesties duomenis P ir T atitinkamai.
4. Išskirkime iš įvesties P ir išvesties T duomenų rinkinių fragmentus, turinčius po 200 pradžioje esamų duomenų – taip vadinamą apmokymo duomenų rinkinį. Remiantis šiuo rinkiniu apskaičiuosime optimalias neurono svorio koeficientų reikšmes. Likę duomenys bus panaudoti modeliui verifikuoti. Tuomet, panaudojant jau esamas P ir T matricas, apibrėžkime dvi naujas – Pu ir Tu, kurios turės pirmus 200 duomenų.
5. Sukursime tiesinės regresijos modelį panaudojant apmokymo duomenų matricas Pu ir Tu.
6. Sukurti prognozės klaidos vektorių e. Nubraižyti prognozės klaidos grafiką. Aprašyti jo ašis ir suteikti pavadinimą.
7. Nubraižyti prognozės klaidų histogramą.
8. Apskaičiuoti vidutinės kvadratinės prognozės klaidos reikšmę.

Palyginti skirtumus tarp MSE ir MAD įverčių.

1. Scenarijų išsaugokite nauju vardu (padarykite sukurtos programos kopiją). Eksperimento būdu parinkite mokymosi greičio lr reikšmę (0< lr <= 1). Parenkama reikšmė (pvz. 0.1), ji naudojama modelyje (žr. sekantį žingsnį). Jei modelis nekonverguoja, parinkta reikšmė mažinama (pvz. 10 kartų) ir procesas kartojamas kol modelis nepradės konverguoti.
2. Sukurti tiesinį neuroną.
3. Apibrėžti siekiamą mokymosi klaidos MSE reikšmę intervale 150 – 300 ir maksimalų epochų kiekį.
4. Atspausdinti gautas po apmokymo svorio koeficientų reikšmes
5. Pakeisti naudojamos eilę ir stebėti modelio kitimą. (kai n=6 ir kai n=10).

* Antra dalis

Iš pirmo laboratorinio darbo duomenų rinkinio, reikia padaryti prognozavimo modelį. Ir atlikti apačioje paminėtus uždavinius, kurie padės patikrinti modelio veikimą.

Darbo eiga:

1. Sukurti reikšmės prognozavimo modelį.
2. Įvertinti sukurto modelio vidutinį tikslumo įvertį, taikant 10 intervalų kryžminės patikros metodą.
3. Duomenų rinkiniai

* Pirma dalis:

Pirmoje dalyje naudosime duomenų rinkinį saulės dėmių aktyvumas. Duomenų rinkinį sudaro 2 atributai:

* Metai – kuriais metais buvo stebėtos saulės dėmės.
* Dėmių kiekis – skaičius nurodantis kiek buvo užregistruota saulės dėmių.

Mūsų laboratoriniame darbe naudosime šiuos atributus prognozuoti ateinančių metų saulės dėmių kiekius.

* Antra dalis:

Antroje dalyje buvo pasirinktas duomenų rinkinys pavadinimu: Video Game Sales with Ratings.

Šis duomenų rinkinys buvo naudotas pirmame laboratoriniame darbe.

<https://www.kaggle.com/datasets/rush4ratio/video-game-sales-with-ratings?select=Video_Games_Sales_as_at_22_Dec_2016.csv>

Duomenų rinkinį sudaro, 16 stulpelių (atributų) ir 16719 įrašai. Jame yra 6 kategoriniai atributai ir 10 tolydiniai atributai.

Duomenų rinkinio atributai:

* Name – Žaidimo pavadinimas.
* Platform- Platforma kurioje buvo išleistas žaidimas.
* Year\_of\_Release – metai, kuriais buvo išleistas žaidimas.
* Genre – žaidimo žanras.
* Publisher – Firma, kuri išleido šį žaidimą.
* NA\_Sales – Žaidimo pardavimai Šiaurės Amerikoje (milijonais).
* EU\_Sales - Žaidimo pardavimai Europoje (milijonais).
* JP\_Sales - Žaidimo pardavimai Japonijoje (milijonais).
* Other\_Sales - Žaidimo pardavimai kituose žemynuose (milijonais).
* Global\_Sales – totalus parduotų kopijų skaičius visame pasaulyje (milijonais).
* Critic\_Score – Kritikų vertinimas (nuo 0 iki 100 taškų).
* Critic\_Count – Kritikų skaičius.
* User\_Score – Naudotojų vertinimas (nuo 0 iki 100 taškų).
* User\_Count – Naudotojų skaičius.
* Developer – Firma, kuri sukūrė žaidimą.
* Rating – ESRB vertinimas (<https://www.esrb.org/ratings-guide/>)

Prognozavimo modeliui naudosime:

* Name – Žaidimo pavadinimas.
* Platform- Platforma kurioje buvo išleistas žaidimas.
* Year\_of\_Release – metai, kuriais buvo išleistas žaidimas.
* Genre – žaidimo žanras.
* Publisher – Firma, kuri išleido šį žaidimą.
* NA\_Sales – Žaidimo pardavimai Šiaurės Amerikoje (milijonais).
* EU\_Sales - Žaidimo pardavimai Europoje (milijonais).
* JP\_Sales - Žaidimo pardavimai Japonijoje (milijonais).
* Other\_Sales - Žaidimo pardavimai kituose žemynuose (milijonais).
* Global\_Sales – totalus parduotų kopijų skaičius visame pasaulyje (milijonais).

Bandysime prognozuoti: Global\_Sales – totalus parduotų kopijų skaičius visame pasaulyje (milijonais).

Lentelė 1 parodo kategorinio tipo atributų skaičiavimus, duomenų rinkinio analizei po modifikavimo.

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Atributo pavadinimas** | **Kiekis** | **Trūkstams reikšmės** | **Kardinalumas** | **Minimali reikšmė** | **Maksimali reikšmė** | **1-asis kvartilis** | **3-iasis kvartilis** | **Vidurkis** | **Mediana** | **Standartinis nuokrypis** |
| Year\_of\_Release | 16416 | 0 | 39 | 1980 | 2020 | 2003 | 2010 | 2006,4899 | 2007 | 5,8811 |
| NA\_Sales | 16416 | 0 | 401 | 0 | 41,36 | 0 | 0,24 | 0,2641 | 0,08 | 0,8190 |
| EU\_Sales | 16416 | 0 | 307 | 0 | 28,96 | 0 | 0,11 | 0,146 | 0,02 | 0,5071 |
| JP\_Sales | 16416 | 0 | 244 | 0 | 10,22 | 0 | 0,04 | 0,0786 | 0 | 0,3113 |
| Other\_Sales | 16416 | 0 | 155 | 0 | 10,57 | 0 | 0,03 | 0,0477 | 0,01 | 0,1882 |
| Global\_Sales | 16416 | 0 | 628 | 0,01 | 82,53 | 0,06 | 0,47 | 0,5367 | 0,17 | 1,5599 |

.

lentelė 2 parodo kategorinio tipo atributų skaičiavimus, duomenų rinkinio analizei po modifikavimo

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Atributo pavadinimas** | **Name** | **Platform** | **Genre** | **Publisher** |
| **Kiekis** | 16416 | 16416 | 16416 | 16416 |
| **Trūkstamos reikšmės** | 0 | 0 | 0 | 0 |
| **Kardinalumas** | 11397 | 31 | 12 | 579 |
| **Moda** | Need for Speed: Most Wanted | PS2 | Action | Electronic Arts |
| **Modos dažnumas** | 12 | 2127 | 3307 | 1344 |
| **Moda, %** | 0,073% | 12,957% | 20,145% | 8,187% |
| **2-oji Moda** | Ratatoille | DS | Sports | Activision |
| **2-osios modos dažnumas** | 9 | 2122 | 2306 | 976 |
| **2-oji moda, %** | 0,055% | 12,926% | 14,047% | 5,945% |

1. Funkcinių reikalavimų vykdymas

* Pirma dalis:

Pirmoje dalyje, kursime prognozavimo modeli, naudojant tiesinį neuroną. Šias neuronas bus apmokytas duomenimis iš duomenų failo sunspot.txt. Prognozavimui bus naudojama antra eilė n=2, tai reiškia, kad prognozuojant duomenis bus naudojamos ankstesnės reikšmės. a(k-1), a(k-2), ..., a(k-n).

Taigi prieš pradedant kurti neuroną, mums reikia išsiaiškinti kokie yra duomenys, juos atvaizduoti grafiku ir suskirstyti į matricas P it T, kurios bus naudojamos apmokant mūsų prognozavimo modelį.

Duomenų atvaizdavimas

Pirmiausiai reikia atvaizduoti duotus duomenis mūsų duomenų rinkinyje. Pagal atvaizduotus duomenis lengva matyti, jog nuspėti saulės dėmių aktyvumą nėra lengva, nes grafikas turi ir metų kuriais saulės aktyvumas būna labai didelis ir tokių metų kuriais būna labai mažas.

Paveikslėlis, kuriame yra tekstas, ekrano kopija, Grafikas, Šriftas

Automatiškai sugeneruotas aprašymas

pav. 1 Saulės dėmių grafikas nuo 1700m. iki 2014m.

Toliau reikia nubrėžti trimatę diagramą, kurioje yra atvaizduoti įvesties ir išvesties duomenys - P ir T matricos.

Paveikslėlis, kuriame yra diagrama, ekrano kopija

Automatiškai sugeneruotas aprašymas

pav. 2 P ir T matricų atvaizdavimas trimatėje erdvėje

Tiesinės autoregresijos modelis, tiesinis neuronas

Išskirkime iš įvesties P ir išvesties T duomenų rinkinių fragmentus, turinčius po 200 pradžioje esamų duomenų – taip vadinamą apmokymo duomenų rinkinį. Remiantis šiuo rinkiniu apskaičiuosime optimalias neurono svorio koeficientų reikšmes (autoregresinio modelio parametrus). Apskaičiuoti svorių koeficientai pateikti apačioje.

|  |
| --- |
| **Kodas:**  # Extract the first 200 data points for training  Pu = X\_train[:200]  Tu = y\_train[:200]  # Calculate optimal weight coefficients for the training dataset  optimal\_weights = calculate\_optimal\_weights(Pu, Tu) |
| **Optimal Coefficients:**  w1: -0.6513238762657068  w2: 1.4272601056966459  b: 9.062449540429913 |

Toliau reikia sukurti tiesinės autoregresijos modelį naudojant duomenų matricas Pu ir Tu.

|  |
| --- |
| **Kodas:**  # Build the Neural Network  class NeuralNetwork:  def \_\_init\_\_(self):  # Initialize weights and bias  self.weights = optimal\_weights  self.bias = 0.0    def forward(self, X):  return np.dot(X, self.weights) + self.bias  # Train the Neural Network  network = NeuralNetwork()  learning\_rate = 0.0001  epochs = 25000  losses = []  for epoch in range(epochs):  # Forward pass  y\_pred = network.forward(X\_train)    # Compute loss (MSE)  loss = np.mean((y\_pred - y\_train) \*\* 2)  losses.append(loss)    # Backpropagation (Gradient Descent)  grad\_weights = np.dot(X\_train.T, (y\_pred - y\_train)) / len(X\_train)  grad\_bias = np.mean(y\_pred - y\_train) |

Modelio verifikacija

Sekančiame žingsnyje atliksime modelio verifikaciją – t.y. patikrinsime prognozavimo kokybę atliekant modelio veikimo imitaciją. Pradžioje tai atliksime su apmokymo duomenų rinkiniu, kuris buvo panaudotas svorio koeficientams apskaičiuoti. Pagal diagramas, galime matyti, jog prognozuotos reikšmės yra labai artimos tikroms reikšmėms, tai parodo, jog prognozavimo modelis yra tikslus.

Paveikslėlis, kuriame yra tekstas, ekrano kopija, Grafikas, Šriftas

Automatiškai sugeneruotas aprašymas

pav. 3 Apskaičiuotų reikšmių palyginimas su tikromis reikšmėmis

Paveikslėlis, kuriame yra tekstas, ekrano kopija, Šriftas, Grafikas

Automatiškai sugeneruotas aprašymas

pav. 4 Apskaičiuotų reikšmių palyginimas su tikromis reikšmėmis (Apmokymo duomenys)

Paveikslėlis, kuriame yra tekstas, ekrano kopija, Šriftas, Grafikas

Automatiškai sugeneruotas aprašymas

pav. 5 Apskaičiuotų reikšmių palyginimas su tikromis reikšmėmis(Testavimo duomenys)

Toliau norint patikrinti ar modelis tikrai teisingas reikia apskaičiuoti prognozės klaidos vektorių e. Atvaizduoti prognozės klaidos grafiką su histograma ir apskaičiuoti vidutinės kvadratinės prognozės klaidos reikšmę MSE, kartu su prognozės absoliutaus nuokrypio medianą MAD. Iš klaidų histogramos, galime pamatyti, jog pasikartojančios klaidos dažniausiai būnu nedidelės, tačiau ir pasitaiko klaidų kai modelis nutolsta daug nuo tikslo.

|  |
| --- |
| **Klaidos vektorius**: [ 5.50569132 1.73404859 -4.53175005 -12.53663581 39.39587613  -7.32379222 8.71925924 2.30857307 10.96729162 8.13363885  7.10847791 7.06244954 0.91696975 -3.54033705 -6.56609024  -4.44207015 8.36761401 14.66465168 -2.35383891 -2.37589867  5.93414375 12.52775108 -10.56681457 -8.12965088 -25.52494763  -27.66421727 29.38492009 3.60872752 -0.833922 -6.40296846  17.40433106 -3.03402497 -6.96581257 -5.35800815 -22.83188889  5.82564515 -31.92215324 13.7310873 7.91876995 7.46872575  -1.3937892 -4.4453034 13.87213371 -5.22243195 -0.49430868  -6.70239077 -8.17627151 -12.35489917 2.59108553 26.76780434  -24.48894712 15.30738855 10.04396676 -4.00146936 5.51916408  -15.17818817 0.22153416 2.72857057 -8.12905081 -22.19165312  29.77341443 -4.88853737 -2.44160216 10.13423891 4.58725221  -26.9154907 -13.86622908 -21.77965036 13.75934941 2.17538933  -6.6859333 16.28031877 -15.62214647 23.51117715 -21.13776988  -59.95161548 -25.22883816 42.28738533 3.59334591 -19.68724988  11.75360725 -2.56442992 6.4878008 -15.64539856 -46.19654669  -20.10673472 12.40090176 -7.94122828 2.15571422 -6.34035835  -13.93027224 4.05935617 -3.93575807 15.96789169 -3.66936717  12.22080983 3.20482815 3.86354671 1.44797478 -8.08791601  2.81943499 8.1441425 -6.87494035 7.56400793 9.74382792  11.6701297 -2.90201794 10.96729162 8.13363885 6.10847791  5.48970965 3.54742619 8.93295143 -13.7717955 3.89801897  10.92007873 7.61921557 1.17597378 7.77697579 9.2668382  4.63208826 8.21222283 0.31167425 3.60514274 -8.53604355  -0.62869255 -7.02220472 0.84090249 -7.99585146 18.75921734  3.3269394 8.76218644 -9.32927804 -35.24508397 -40.05093483  8.06272149 23.56283122 -19.8124545 -0.27954063 8.82050305  -4.46497851 8.21770866 -5.86946233 -16.69321151 -5.61700438  -27.50037896 -15.02088032 26.988899 -2.33606485 -22.8382155  4.1956567 4.79844329 8.55410434 6.63328059 1.37546888  -12.78777717 -15.71586353 -21.4186938 11.40208628 7.85497532  -2.56561444 -0.8811427 -13.56621451 16.48542395 6.69529063  4.70757107 -29.36791174 -15.26093358 -49.07060994 48.25363739  -25.04569753 16.34603006 -8.17341886 13.30177846 -6.98370309  1.68980481 16.02500817 -0.47165666 -16.32796145 -3.17317033  5.29213121 -4.47343344 -2.67233627 6.72236822 16.59524696  -2.12488938 4.33373401 4.58605989 6.06674304 -20.85467298  -17.11545379 4.80477771 4.83291556 1.0262083 7.60383396  1.3226459 -8.18439051 18.66405161 -1.39617385 12.51916408  1.83099109 -9.75522156 -1.9399273 -10.62439905 19.0514935  -17.55023283 13.38108678 -5.38388561 20.94702425 1.52214099  1.25085653 9.86354671 -2.11558586 -24.31627328 12.63043575  -44.19594662 39.37203959 -7.06716503 9.64986233 -4.38639452  7.42090499 6.10967024 -8.49252409 -14.58207192 -3.21061171  2.74884575 -12.14133125 10.44739032 15.03109406 -2.89223861  4.58725221 5.0845093 1.46144754 -18.00015277 -25.41810154  -14.20440155 9.66419149 2.81013927 -3.55702744 0.14831174  2.28091103 6.04396676 1.70757107 -20.08613142 -43.35120573  -31.69604855 29.43286511 -30.83140528 29.16251664 -27.97642488  20.83219123 -4.20657454 4.20172698 -32.3470443 -81.30696195  -3.01692275 2.75387919 -9.6459674 3.50188924 11.35508505  -24.81377889 0.12684424 14.2754252 -9.90201794 -23.04188764  -27.62618364 6.61267729 -6.87242362 -13.68831014 22.88442975  -32.70013039 25.90469712 -16.64301391 18.26624594 -3.89772444  -10.80435111 -52.44147789 -31.43942917 14.71464543 -11.6674349  -6.64907638 15.78795525 -16.86469302 13.07771469 -8.20776687  -13.10699886 -58.01421779 -25.0999323 26.43715861 -35.74852779  30.30311067 -6.86838646 -3.66268902 -1.94256048 6.21341515  -11.81603928 -29.39974302 -6.92202897 -18.88708871 7.33328163  -14.01922 21.20055027 -7.33058683 -5.53187431 10.82729766  3.93163484 6.28341214 4.78496272 -3.18248167 -27.70668427  21.99509354 -10.28192528 -14.94242841] |

Paveikslėlis, kuriame yra tekstas, ekrano kopija, Grafikas, diagrama

Automatiškai sugeneruotas aprašymas

pav. 6 Prognozės klaidos grafikas

Paveikslėlis, kuriame yra tekstas, diagrama, ekrano kopija, Grafikas

Automatiškai sugeneruotas aprašymas

Histograma 1Prognozės klaidų histograma

|  |
| --- |
| Mean Squared Error (MSE) for Training Data: 351.8303377501935  Mean Squared Error (MSE) for Testing Data: 485.03112284367927  Median Absolute Deviation (MAD) for Training Data: 9.82746125147763  Median Absolute Deviation (MAD) for Testing Data: 12.628882132813253 |

Antrasis scenarijus

Scenarijų išsaugosime nauju vardu. Eksperimento būdu parinksime mokymosi greičio lr reikšmę (0< lr <= 1). Parenkama reikšmė (pvz. 0.1), ji naudojama modelyje (žr. sekantį žingsnį). Jei modelis nekonverguoja, parinkta reikšmė mažinama (pvz. 10 kartų) ir procesas kartojamas kol modelis nepradės konverguoti. Šis mokymosi greitis naudojamas neurono sudarymui.

Toliau reikia atvaizduoti mokymosi klaidos MSE reikšmę intervale 150-300 maksimalų epochų kiekį. Iš grafiko lengva pastebėti, jog nuo 150 epochos, MSE reikšmė beveik nesikeičia, todėl tęsti epochų didinimą nėra labai verta.

Paveikslėlis, kuriame yra tekstas, ekrano kopija, diagrama, linija

Automatiškai sugeneruotas aprašymas

pav. 7 MSE kiekvienoje epochoje

Neuronas

|  |
| --- |
| # Build the Neural Network  class NeuralNetwork:  def \_\_init\_\_(self):  # Initialize weights and bias  self.weights = np.random.randn(window\_size) \* 0.01  self.bias = 0.0    def forward(self, X):  return np.dot(X, self.weights) + self.bias  # Train the Neural Network  network = NeuralNetwork()  learning\_rate = 0.0001  epochs = 1000  losses = []  for epoch in range(epochs):  # Forward pass  y\_pred = network.forward(X\_train)    # Compute loss (MSE)  loss = np.mean((y\_pred - y\_train) \*\* 2)  losses.append(loss)    # Backpropagation (Gradient Descent)  grad\_weights = np.dot(X\_train.T, (y\_pred - y\_train)) / len(X\_train)  grad\_bias = np.mean(y\_pred - y\_train)    # Update weights and bias  network.weights -= learning\_rate \* grad\_weights  network.bias -= learning\_rate \* grad\_bias |

Atspausdinti gautas po apmokymo svorio koeficientų reikšmes. Jas palyginti su gautais rezultatais iš pirmojo scenarijaus.

|  |  |
| --- | --- |
| Pirmasis scenarijus | Antrasis scenarijus |
| w1: -0.6513238762657068  w2: 1.4272601056966459  b: 9.062449540429913 | Optimal Coefficients:  w1: -0.6760819763970705  w2: 1.3715093938395844  b: 0.5533335250003097 |

Papildomi klausymai

* Ar mokymosi procesas yra konverguojantis? Jeigu ne, pamąstyti kas gali būti priežastimi ir pakeisti atitinkamą parametrą.

Mokymosi procesas yra konverguojantis, nes einant per epochas keičiasi svoriai iki tol kol pasiekiamas tikslumas. Tą patį galime matyti ir (pav. 7) grafike, kur atvaizduojamas MSE kitimas kiekvienoje epochoje.

* Kokios yra naujos neurono svorių koeficientų reikšmės ?

w1: -0.6513238762657068

w2: 1.4272601056966459

b: 9.062449540429913

* Kokia yra neurono darbo kokybės įverčio MSE ir MAD reikšmės ?

Mean Squared Error (MSE) for Training Data: 351.8290901817074

Mean Squared Error (MSE) for Testing Data: 485.02956031061854

Median Absolute Deviation (MAD) for Training Data: 9.82746970931214

Median Absolute Deviation (MAD) for Testing Data: 12.628850112742569

* Kokia yra maksimali leistina mokymosi proceso greičio koeficiento lr reikšmė, kuri užtikrina proceso konvergenciją?

learning\_rate = 0.0002

Eksperimentas su skirtingomis modelio eilėmis

* Kai n = 6

Pirmiausiai atliksime eksperimentą, kai modelio eilė yra 6. Tai reiškia, jog apskaičiuoti sekančių metų duomenims naudosime 6 ankstesnių metų reikšmes. Iš skaičiavimų grafikai gavosi labai panašūs, skyrėsi tik svorių koeficientai ir MSE ir MAD reikšmės. Modelis mano manymu tapo tikslesnis, nes sumažėjo MSE.

|  |
| --- |
| Optimal Coefficients:  w1: 0.15341920834885692  w2: -0.23942144028947485  b: 0.2250208403252848  Mean Squared Error (MSE) for Training Data: 291.72883567784953  Mean Squared Error (MSE) for Testing Data: 390.3262880364394  Median Absolute Deviation (MAD) for Training Data: 9.514305667747326  Median Absolute Deviation (MAD) for Testing Data: 12.583422489926157 |

Paveikslėlis, kuriame yra tekstas, ekrano kopija, Šriftas, Grafikas

Automatiškai sugeneruotas aprašymas

Paveikslėlis, kuriame yra tekstas, ekrano kopija, linija, diagrama

Automatiškai sugeneruotas aprašymas

Paveikslėlis, kuriame yra tekstas, ekrano kopija, Grafikas, diagrama

Automatiškai sugeneruotas aprašymas

Paveikslėlis, kuriame yra diagrama, Grafikas, tekstas, ekrano kopija

Automatiškai sugeneruotas aprašymas

Paveikslėlis, kuriame yra tekstas, ekrano kopija, Grafikas, linija

Automatiškai sugeneruotas aprašymas

Paveikslėlis, kuriame yra tekstas, ekrano kopija, Šriftas, Grafikas

Automatiškai sugeneruotas aprašymas

Paveikslėlis, kuriame yra tekstas, ekrano kopija, Šriftas, Grafikas

Automatiškai sugeneruotas aprašymas

* Antra dalis:

Šioje dalyje naudosime duomenų rinkinį iš pirmo laboratorinio darbo. Rinkinį sutvarkysime, kad nebūtų pradingusių arba netaisyklingu duomenų, ir naudosime jį apskaičiuoti vieną jo reikšmę, pasitelkiant kitas reikšmes.

Reikšmė kurią bandysime surasti: **Global\_Sales** – totalus parduotų kopijų skaičius visame pasaulyje (milijonais).

Šiai reikšmei surasti naudosime:

* Platform- Platforma kurioje buvo išleistas žaidimas.
* Year\_of\_Release – metai, kuriais buvo išleistas žaidimas.
* Genre – žaidimo žanras.
* NA\_Sales – Žaidimo pardavimai Šiaurės Amerikoje (milijonais).
* EU\_Sales - Žaidimo pardavimai Europoje (milijonais).
* JP\_Sales - Žaidimo pardavimai Japonijoje (milijonais).
* Other\_Sales - Žaidimo pardavimai kituose žemynuose (milijonais).

Reikšmės prognozavimo modelis

|  |
| --- |
| import os  import numpy as np  import pandas as pd  import scipy.sparse as sp  from sklearn.model\_selection import KFold  from sklearn.preprocessing import StandardScaler, OneHotEncoder  from sklearn.compose import ColumnTransformer  import tensorflow as tf  from tensorflow.keras.models import Sequential  from tensorflow.keras.layers import Dense  from tensorflow.keras.optimizers import Adam  # Set TensorFlow to use only the CPU  os.environ['CUDA\_VISIBLE\_DEVICES'] = '-1'  # Load the data  data = pd.read\_csv('Video\_Games\_Sales\_as\_at\_22\_Dec\_2016.csv')  data['Year\_of\_Release'] = data['Year\_of\_Release'].fillna(data['Year\_of\_Release'].median()) # Correctly using inplace modification  # Prepare the data  features = ['Platform', 'Year\_of\_Release', 'Genre', 'NA\_Sales', 'EU\_Sales', 'JP\_Sales', 'Other\_Sales']  numeric\_features = ['Year\_of\_Release', 'NA\_Sales', 'EU\_Sales', 'JP\_Sales', 'Other\_Sales']  categorical\_features = ['Platform', 'Genre']  numeric\_transformer = StandardScaler()  categorical\_transformer = OneHotEncoder(handle\_unknown='ignore', sparse\_output=False)  preprocessor = ColumnTransformer(transformers=[  ('num', numeric\_transformer, numeric\_features),  ('cat', categorical\_transformer, categorical\_features)  ])  X = data[features]  y = (data['Global\_Sales'] > 1).astype(int) # Example binary target  X\_transformed = preprocessor.fit\_transform(X)  # K-Fold Cross-Validation  kf = KFold(n\_splits=10, shuffle=True, random\_state=0)  accuracies = []  for train\_index, test\_index in kf.split(X\_transformed):  X\_train, X\_test = X\_transformed[train\_index], X\_transformed[test\_index]  y\_train, y\_test = y[train\_index], y[test\_index]  # Convert sparse matrix to dense, if necessary  X\_train = X\_train.toarray() if sp.issparse(X\_train) else X\_train  X\_test = X\_test.toarray() if sp.issparse(X\_test) else X\_test  # Build the model  model = Sequential()  model.add(Dense(X\_train.shape[1], activation='selu', input\_dim=X\_train.shape[1]))  model.add(Dense(1, activation='sigmoid'))  model.compile(optimizer='adam', loss='binary\_crossentropy', metrics=['accuracy'])  # Train the model  model.fit(X\_train, y\_train, batch\_size=10, epochs=10, verbose=1)  # Evaluate the model  \_, accuracy = model.evaluate(X\_test, y\_test, verbose=0)  accuracies.append(accuracy)  # Calculate the average and standard deviation of the accuracies  average\_accuracy = np.mean(accuracies)  std\_dev\_accuracy = np.std(accuracies)  print(f"Average Accuracy: {average\_accuracy:.2%}")  print(f"Standard Deviation of Accuracy: {std\_dev\_accuracy:.2%}") |

Modelio patikrinimui naudosime 10 intervalų kryžminės patikros metodą. Iš šio metodo galime matyti, jog modelio tikslumas yra labai didelis. Tai yra greičiausiai, dėl to, jog atributas kurio ieškome susideda iš kitų atributų, kuriuos žinome, tai padaro labai aukštą tikslumą.

|  |
| --- |
| Epoch 1/10  1478/1478 [==============================] - 2s 901us/step - loss: 0.0700 - accuracy: 0.9777  Epoch 2/10  1478/1478 [==============================] - 1s 894us/step - loss: 0.0165 - accuracy: 0.9959  Epoch 3/10  1478/1478 [==============================] - 1s 897us/step - loss: 0.0115 - accuracy: 0.9970  Epoch 4/10  1478/1478 [==============================] - 1s 943us/step - loss: 0.0087 - accuracy: 0.9980  Epoch 5/10  1478/1478 [==============================] - 1s 896us/step - loss: 0.0073 - accuracy: 0.9977  Epoch 6/10  1478/1478 [==============================] - 1s 875us/step - loss: 0.0065 - accuracy: 0.9982  .......  .......  Average Accuracy: 99.79%  Standard Deviation of Accuracy: 0.10% |

1. Išvados

* Laboratorinis darbą atlikti pavyko.
* Pirmasis modelis, kuriame naudojome vieną neuroną prognozuoti ateinančių metų duomenis, buvo gan tikslus.
* Antrasis modelis, kuriame prognozavome pirmojo laboratorinio darbo duomenų rinkinį, pasiekė net 99,79% tikslumą.