Logo

Description automatically generated

VILNIAUS UNIVERSITETAS MATEMATIKOS IR INFORMATIKOS FAKULTETAS

INFORMACINIŲ SISTEMŲ INŽINERIJA 3 KURSAS

**Perceptrono mokymas**

Laboratorinis darbas

Atliko: Deividas Gelžinis

VU el. p.: deividas.gelzinis@mif.stud.vu.lt

Vadovas: Viktor Medvedev

Vilnius

2021

Turinys

[1. Tikslas 3](#_Toc117005554)

[2. Įrankiai 3](#_Toc117005555)

[3. Darbo eiga 3](#_Toc117005556)

[3.1 Duomenų gavimas 3](#_Toc117005557)

[3.2 Duomenų paruošimas 3](#_Toc117005558)

[3.3 Svorių generavimas 4](#_Toc117005559)

[3.4 Neurono mokymas 4](#_Toc117005560)

[4. Rezultatai 4](#_Toc117005561)

[5. Išvados 6](#_Toc117005562)

[6. Priedas 6](#_Toc117005563)

# Tikslas

Apmokyti vieną neuroną spręsti dviejų klasių uždavinį, atlikti tyrimą su dviem

duomenų aibėm.

# Įrankiai

Neurono programiniam kodui buvo naudojama „Python“ kalba, o grafikų atvaizdavimui „Python“ kalbos biblioteka „Matplotlib“.

# Darbo eiga

## Duomenų gavimas

Darbui naudojome 2 duomenų rinkinius:

* Irisų duomenų aibė (<https://archive.ics.uci.edu/ml/datasets/iris>)
* Krūties vėžio duomenų aibė (<https://archive.ics.uci.edu/ml/datasets/Breast+Cancer+Wisconsin+(Diagnostic)>

## Duomenų paruošimas

Irisų duomenų aibei reikėjo pašalinti „Iris-setosa“ duomenis bei paversti „Iris-versicolor“ ir „Iris-virginica“ klases į 1 arba 0. Savo programos kode aprašiau „Iris-versicolor“ kaip 0 ir „Iris-virginica“ kaip 1.

Krūties vėžio duomenų aibėje reikėjo pašalinti duomenų eilutes turinčias „?“, pirmajį įvesties duomenų aibėje stulpeli ir paversti klases „2” ir „4” į 1 arba 0. Savo programos kode aprašiau „2“ kaip 0 ir „4“ kaip 1.

Irisų duomenų aibėje pirmieji 50 įrašų buvo „Iris-versicolor“, todėl naudojau Python bibliotekos „Random“ funkcija „shuffle“, kuris atsitiktinai sudėlioja gauta sąrašą. Tas pats buvo atlikta vėžio duomenims, jog išvengti galimo duomenų pasikartojimo.

Duomenis buvo išskaidomi santykiu 80:20, 80% duomenų buvo apmokomieji, o likusieji 20% skirti testavimui.

Po duomenų sutvarkymo Irisų duomenų aibėje liko 100 įrašu, o krūties vėžio duomenų aibėje 683.

## Svorių generavimas

Pradiniams svoriams generuoti buvo naudojama „Python“ biblioteka „Random“ ir „Uniform“ funkcija.

Irisų duomenų aibei reikėjo sugeneruoti w0 ir dar 4 atsitiktinius svorius.

Krūties vėžio duomenų aibei reikėjo w0 ir dar 9 atsitiktinių svorių.

## Neurono mokymas

Programa paleidus mes pasirenkame norima aktyvacijos funkcija ir norima duomenų aibe. Tada programa pagal mūsų nustatyta epochos skaičių atlieka neurono mokymą.

Mokymosi epocha – tai neuronų mokymosi proceso dalis, kurios metu apdorojamas visas įėjimų vektorių rinkinys vieną kartą.

Programa naudoja formule:

Ir naudodama *f(a)* (mūsų pasirinkta aktyvacijos funkcija) išveda gauta klase.

Toliau apmokymui mes naudojame paklaidos formule:

O tikslumui apskaičiuoti mes tiesiog patikriname kiek teisingų duomenų būna palyginus su visais duomenimis.

Gautai klasei neatitikus norimai klasei mes naudojame ADELINE formule:

*t –* tai iteracijos skaičius, *w* – svoris, *η* – mokymosi greitis, *x* – įvestis.

# Rezultatai

Lentelė (1 lentelė) parodo gauta tikslumą bei paklaida naudojant vėžio ir irisų duomenų aibes po skirtingų epochų kiekio. Mokymosi greitis 0,01

1 lentelė. Tikslumo ir paklaidos priklausomumas nuo epochų skaičiaus

|  |  |  |  |
| --- | --- | --- | --- |
| **Rinkinys** | **Aktyvacija** | **Tikslumas** | **Paklaida** |
| **Epochų kiekis: 1** | | | |
| **Irisai** | Slenkstinė | 45 | 5,5 |
| **Vėžys** | Slenkstinė | 70,8 | 20 |
| **Irisai** | Sigmoidinė | 0 | 2,83 |
| **Vėžys** | Sigmoidinė | 53,23 | 9,86 |
| **Epochų kiekis: 10** | | | |
| **Irisai** | Slenkstinė | 90 | 1,0 |
| **Vėžys** | Slenkstinė | 95,46 | 3,0 |
| **Irisai** | Sigmoidinė | 30 | 1,54 |
| **Vėžys** | Sigmoidinė | 74,14 | 4,14 |
| **Epochų kiekis: 100** | | | |
| **Irisai** | Slenkstinė | 100 | 0 |
| **Vėžys** | Slenkstinė | 98,5 | 1,0 |
| **Irisai** | Sigmoidinė | 80 | 0,38 |
| **Vėžys** | Sigmoidinė | 94,4 | 1,46 |

Chart

Description automatically generatedChart, histogram

Description automatically generated

paveikslas. Tikslumo ir paklaidos kitimas nuo epochos skaičiaus naudojant irisų duomenų aibę

Chart, line chart

Description automatically generatedChart, histogram

Description automatically generated

paveikslas. Tikslumo ir paklaidos kitimas nuo epochos skaičiaus naudojant irisų duomenų aibe

1 paveikslas ir 2 paveikslas parodo kaip keičiasi tikslumas ir paklaida naudojant irisų duomenų aibę. Žemiau parodyta kaip keičiasi tos pačios tikslumo ir paklaidos vertes keičiantis mokymosi greičiui. Pavyzdžiams naudojama irisų duomenų aibė:

Chart, line chart

Description automatically generatedChart, histogram

Description automatically generated

paveikslas. Tikslumas ir paklaida naudojant 0,1 mokymo greitį sigmoidinei funkcijai

Chart, line chart

Description automatically generatedChart, line chart

Description automatically generated

4 paveikslas. Tikslumas ir paklaida naudojant 0,001 mokymo greitį sigmoidinei funkcijai

# Išvados

* Slenkstine aktyvacijos funkcija yra mažiau tiksli ir pastovi, todėl ji netinka didesnių duomenų apmokymui ir labai tikslių rezultatų gavimui.
* Didėjant epochų skaičiui iki tam tikro epochų kiekio tikslumas auga, bet modelis gali persimokyti taip sumažinant jo tikslumą.
* Tikslumui kylant paklaida mažėja.
* Mokymosi greičio dydis nulemia kiek epochų reikės, jog pasiektu aukščiausia tikslumą bei ko mažesnis mokymosi greitis tuo stabilesnis apmokymas.

# Priedas

1. import random
2. from random import uniform, shuffle
3. import numpy as np
4. import matplotlib.pyplot as plt
5. # Aprašomi naudojami masyvai
6. irisResult = []
7. irisData = []
8. vezioResult = []
9. vezioData = []
10. weights = []
11. learningSet = []
12. learningSetResults = []
13. learningAccuracy = []
14. learningErrors = []
15. testAccuracy = []
16. testErrors = []
17. epochs = 100
18. learningRate = 0.001
19. # Nuskaito ir sutvarko Irisu duomenis
20. def irisuSkaitymas():
21. with open('iris.data', 'r') as f:
22. i = 0
23. for line in f.read().split("\n"):
24. isWrong = 0
25. irisTemp= []
26. irisTemp.append(1)
27. for x in line.split(","):
28. if x == 'Iris-setosa' or not x:
29. isWrong = 1
30. else:
31. if x == 'Iris-versicolor':
32. irisTemp.append(0)
33. elif x == 'Iris-virginica':
34. irisTemp.append(1)
35. else:
36. irisTemp.append(float(x))
37. if not isWrong:
38. irisData.append(irisTemp)
39. i =i + 1
40. # Nuskaito ir sutvarko Vezio duomenis
41. def vezioSkaitymas():
42. with open('breast-cancer-wisconsin.data', 'r') as f:
43. i = 0
44. for line in f.read().split("\n"):
45. isfirst = 1
46. wrongData = 0
47. vezioTemp = []
48. vezioTemp.append(1)
49. for x in line.split(","):
50. if isfirst:
51. isfirst = 0
52. else:
53. if x != '?':
54. vezioTemp.append(float(x))
55. if x == '?' or not x:
56. wrongData = 1
57. if not wrongData:
58. vezioData.append(vezioTemp)
60. # Išsaugo Irisu klases į atskira masyva
61. def savingIrisResults():
62. shuffle(irisData)
63. print("Irisų duomenu aibių skaičius: "+str(len(irisData)))
64. for data in irisData:
65. irisResult.append(data.pop())
66. # Išsaugo Vežio klases į atskira masyva
67. def savingVezioResults():
68. shuffle(vezioData)
69. print("Vėžio duomenu aibių skaičius: " + str(len(vezioData)))
70. for data in vezioData:
71. newRes = 0
72. if data.pop() == 2:
73. newRes = 0
74. else:
75. newRes = 1
76. vezioResult.append(newRes)
78. # Sugeneruojami pradiniai svoriai naudojant skaicius nuo 0 iki 1
79. def generateWeights(dataChoice):
80. if dataChoice == '1':
81. for \_ in range(5):
82. weights.append(uniform(0, 1))
83. else:
84. for \_ in range(10):
85. weights.append(uniform(0, 1))
86. print(weights)
87. # Aprašoma slenkstine aktivacija
88. def slenkstinisActivation(result):
89. if result>=0:
90. return 1
91. else:
92. return 0
93. # Aprašoma sigmoidine aktivacija apvalinama nuo 0.7 ir 0.3
94. def sigmoidinisActivation(result):
95. sigmoid = 1/(1+np.exp(-result))
96. if sigmoid>=0.7:
97. return 1
98. elif sigmoid<=0.3:
99. return 0
100. else:
101. return sigmoid
102. # Aprašoma adeline funkcija svoriu atnaujinimui
103. def adeline(w, t, y, x):
104. for i in range(0, len(w)):
105. w[i] = w[i] + learningRate \* (t - y) \* x[i]
106. return w
107. # Nupiešiami tikslumo ir paklaidos duomenis pagal epocha
108. def drawResults(learningAccuracy, learningError):
110. plt.plot(range(0,len(learningAccuracy)), learningAccuracy)
111. # naming the x axis
112. plt.xlabel('Epochos')
113. # naming the y axis
114. plt.ylabel('Tikslumas')
116. plt.show()
118. plt.plot(range(0,len(learningError)), learningError)
119. # naming the x axis
120. plt.xlabel('Epochos')
121. # naming the y axis
122. plt.ylabel('Paklaida')
124. plt.show()
125. # Testuojamas neuronas su galutiniais svoriais
126. def neuronTest(weights, testSet, activationChoice, testSetResults):
127. correct = 0
128. error = 0
129. for i in range(len(testSet)):
130. a = np.dot(testSet[i], weights)
131. if activationChoice == '1':
132. y = slenkstinisActivation(a)
133. if y == testSetResults[i]:
134. correct += 1
135. error += pow(y - testSetResults[i], 2)
136. elif activationChoice == '2':
137. y = sigmoidinisActivation(a)
138. if y == testSetResults[i]:
139. correct += 1
140. error += pow(y - testSetResults[i], 2)
142. accuracy = 100 / len(testSet) \* correct
143. testAccuracy.append(accuracy)
144. testErrors.append(error/2)
145. print(testAccuracy)
146. print(testErrors)
147. # Apmokomas neuronas naudojant adeline funkcija bei išskaidom duomenis santykiu 80/30
148. def neuronLearning(data, weights, results, activationChoice):
149. # Skaidomi duomenis
150. learnSetSize = int(len(data) \* 80 / 100)
151. learningSet = data[0:learnSetSize]
152. learningSetResults = results[0:learnSetSize]
153. testSet = data[learnSetSize:len(data)]
154. testSetResults = results[learnSetSize:len(results)]
155. # Pradedamos epochos
156. for \_ in range(epochs):
157. correct = 0
158. error = 0
159. # Vykdomos iteracijos
160. for i in range(len(learningSet)):
161. # Atlieka 2 vektoriu apjungima i bendra suma
162. a = np.dot(learningSet[i], weights)
163. if activationChoice == '1':
164. y = slenkstinisActivation(a)
165. if y == learningSetResults[i]:
166. correct += 1
167. else:
168. weights = adeline(weights, learningSetResults[i], y, learningSet[i])
169. error += pow(y - learningSetResults[i], 2)
170. elif activationChoice == '2':
171. y = sigmoidinisActivation(a)
172. if y == learningSetResults[i]:
173. correct += 1
174. else:
175. weights = adeline(weights, learningSetResults[i], y, learningSet[i])
176. # Atlieka paklaidos skaičiavima pakeliant gauta rezulta kvadratu
177. error += pow((1/(1+np.exp(-a))) - learningSetResults[i], 2)
178. # Aprašo epochos tiksluma
179. learningAccuracy.append(100 / len(learningSet) \* correct)
180. # Aprašo epochos paklaida
181. learningErrors.append(error / 2)
182. drawResults(learningAccuracy, learningErrors)
183. neuronTest(weights, testSet, activationChoice, testSetResults)
184. # Pagrindine veikimo funkcija
185. def main():
186. # Pasirenkame norima duomenu rinkini
187. dataChoice = input("Choose Irisu(1) or Vezio(2) data:")
188. if dataChoice == "1":
189. irisuSkaitymas()
190. savingIrisResults()
191. generateWeights(dataChoice)
192. data = irisData
193. results = irisResult
194. elif dataChoice == "2":
195. vezioSkaitymas()
196. savingVezioResults()
197. generateWeights(dataChoice)
198. data = vezioData
199. results = vezioResult
200. else:
201. print('Wrong choice selection')
202. # Pasirenkame norima aktivacijos funkcija
203. activationChoice = input("Choose Slenkstine(1) or Sigmoidine(2) Activation:")
204. if dataChoice == '1' or dataChoice == '2':
205. neuronLearning(data, weights, results, activationChoice)
206. else:
207. print('Wrong choice selection')
208. main()