VILNIAUS UNIVERSITETAS
MATEMATIKOS IR INFORMATIKOS FAKULTETAS
INFORMATIKOS INSTITUTAS
KOMPIUTERINIO IR DUOMENŲ MODELIAVIMO KATEDRA

PRAKTINĖ UŽDUOTIS NR. 2

Atliko:
Matas Valiūnas

Vilnius
2022

Duomenys

Dirbtinio neurono apmokymui buvo naudojami 2 duomenų failai:

- ***iris.data***: duomenys apie vilkdalgį. 150 eilučių, iš kurių buvo panaudota 100 eilučių (66%), kadangi vienos klasės duomenys buvo pašalinti, kad būtų tik 2 klasės. Vienoje eilutėje yra 4 požymiai ir klasė - *Iris-versicolor* ir *Iris-virginica*, kurios atitinkamai buvo paverstos į *0* ir *1*.
- ***breast-cancer-winsconsin.data***: duomenys apie krūties vėžį. 699 eilutės, iš kurių buvo panaudota 683 eilučių (98%), kadangi kai kurios eilutės buvo pašalintos dėl nepilnų požymių. Vienoje eilutėje yra 9 požymiai ir klasė - *2* (nepiktybinis) ir *4* (piktybinis), kurie atitinkamai buvo paversti į *0* ir *1*.

Kintamieji

trainingDataSize (treniravimo duomenų dydis) kintamasis nustato kiek duomenų yra skiriama mokymui (pvz. jei *trainingDataSize* = 0.8, tai 80% duomenų bus skiriama mokymuisi ir likę 20% - testavimui).

```
trainingDataSize = 0.8                                  # How much of the data is assigned for training (0.8 = 80% for training / 20% for testing)
TrainingData, TestingData = Data[:int(len(Data) * trainingDataSize)], Data[int(len(Data) * trainingDataSize):]      # Assigns data to training and testing
```

pav. 1 Kintamasis trainingDataSize

learningRate (mokymosi greitis) kintamasis nustato kokio dydžio pokyčiai bus taikomi svoriams ir kaip greitai jie dėl to kis.
generations (epochos) kintamasis nustato kiek ciklų neuronas mokysis iš mokymosi duomenų.

```
learningRate = 0.1              # How fast the weights change (0, 1]
generations = 10                # How many cycles the learning happens through the same data
```

pav. 2 Kintamieji learningRate ir generations

Weights (svoriai) yra svorių sąrašas prasidedantis svoriais ir paskutinis elementas yra neurono nusistatymas (bias). Programos pradžioje yra sukuriamas sąrašas iš tiek reikšmių, kiek yra požymių ir papildomai vienos dėl neurono nusistatymo (bias). Jų visų reikšmės yra sugeneruojamos (0, 1) intervale su *random.random()* funkcija.

```
# Weights [W₁, W₂, ..., W₀], W₀ being bias
Weights = [random.random() for w in range(len(Data[0].Attributes) + 1)]
```

pav. 3 Sąrašas Weights

Programos kodas

Klasė *DataLine* skirta talpinti vienai duomenų eilutei. *DataLine.Attributes* yra sąrašas, talpinantis požymius, ir *DataLine.label* yra kintamasis (0 arba 1) talpinantis klasę.

```
4   # Class for storing a single line of data
5   # Atributes [x₁, x₂, ...] and label 0/1
6   class DataLine:
7       def __init__(self, Attributes, label):
8           self.Attributes = Attributes
9           self.label = label
```

pav. 4 Klasė DataLine

Funkcija *FileReading* nuskaito duomenų failą ir gražina sąrašą *DataLine* objektų.
Funkcija *NetInput* gražina požymių ir svorių sandaugos sumą.

```
12  # Reading Data from a file and returning a list
13  def FileReading(fileName):
14      Data = []
15      with open(fileName) as file:
16          for line in file:
17              Data.append(DataLine([float(elem) for elem in line.split(",")[:-1]],  int(line.split(",")[-1])))
18
19      return Data
20
21  # Net input function which is a weighted sum of all the inputs to the neuron
22  def NetInput(Inputs):
23      sum = Weights[-1]
24      for i in range(len(Inputs)):
25          sum += Inputs[i] * Weights[i]
26
27      return sum
```

pav. 5 Funkcijos FileReading ir NetInput

Funkcija *StepFunction* yra slenkstinė funkcija, grąžinanti neurono užsidegimą. Grąžina *1*, jei *NetInput* funkcija duotiems požymiams grąžina teigiamą reikšmę, o jei grąžina neigiamą – grąžina *0*.
Funkcija *SigmoidFunction* yra sigmoidinė funkcija, kuri grąžinanti neurono užsidegimą.

```
29  def StepFunction(Inputs):
30      if NetInput(Inputs) > 0:
31          return 1
32      else:
33          return 0
34
35  def SigmoidFunction(Inputs):
36      return 1 / (1 + pow(math.e, -1 * NetInput(Inputs)))
```

pav. 6 Funkcijos StepFunction ir SigmoidFunction

Funkcija *LossFunction* skaičiuoja ir grąžina paklaidą duotiems duomenims.

```
38  # Function which shows how bad is the neuron
39  def LossFunction(Data):
40      sum = 0
41      for data in Data:
42          sum += pow(ActivasionFunction(data.Attributes) - data.label, 2)
43
44      return sum / 2
```

pav. 7 Funkcija LossFunction

Funkcija *Training* treniruoja neuroną nuolat keisdama svorius duotiems duomenims ir grąžinanti sąrašą kiekvienos epochos paklaidoms.

```
46  # Function which trains a neuron by changing weights and returns a list of loss function results of each epoch
47  def Training(Data):
48      iterations = 0
49      Loss = []
50      while LossFunction(Data) > 0 and iterations < generations:
51          Loss.append(LossFunction(Data))
52          for data in Data:
53              activasion = ActivasionFunction(data.Attributes)
54              if activasion != data.label:
55                  Weights[-1] += learningRate * (data.label - activasion)
56                  for i in range(len(Weights) - 1):
57                      Weights[i] += learningRate * (data.label - activasion) * data.Attributes[i]
58          iterations += 1
59
60      return Loss
```

pav. 8 Funkcija Training

Funkcija *Testing* grąžina kiek procentų neuronas duotiems duomenims nustatė teisingą klasę.

```
62  # Function which tests how many % of data the neuron guessed correctly
63  def Testing(Data):
64      passed = 0
65      for data in Data:
66          if round(ActivasionFunction(data.Attributes)) == data.label:
67              passed += 1
68
69      return str(round(passed / len(Data) * 100, 2)) + "%"
```

pav. 9 Funkcija Testing

Kintamieji *learningRate*, *generations* ir *trainingDataSize* yra aprašyti **Kintamieji** skiltyje.
Eilutėse 77-80 ir 82-85 yra *if* sąlyga leidžianti paleidus programą atitinkamai pasirinkti kokia funkcija bus naudojama neurono aktyvumui nustatyti ir iš kokio failo bus imami duomenys.

```
72  if __name__ == "__main__":
73      learningRate = 0.1          # How fast the weights change (0, 1]
74      generations = 10            # How many cycles the learning happens through the same data
75      trainingDataSize = 0.8      # How much of the data is assigned for training (0.8 = 80% for training / 20% for testing)
76
77      if input("Which function should be used for neuron activation: Step [1] / Sigmoid [2] - ") == "1":
78          ActivasionFunction = StepFunction
79      else:
80          ActivasionFunction = SigmoidFunction
81
82      if input("Which data should be used: Iris [1] / Breast Cancer [2] - ") == "1":
83          Data = FileReading("iris.data")
84      else:
85          Data = FileReading("breast-cancer-wisconsin.data")
```

pav. 10 if sąlygos funkcijos ir duomenų failo pasirinkimui

Funkcija *random.shuffle* išmaišo duomenis, kad skirtingų klasių duomenys butų vienodai pasiskirstę.
89 ir 92 eilutė esantis kodas yra aprašytas **Kintamieji** skiltyje.
94 eilutėje *Loss* yra sąrašas, talpinantis kiekvienos epochos paklaidą.

```
87      random.shuffle(Data)          # Shuffles the data
88
89      TrainingData, TestingData = Data[:int(len(Data) * trainingDataSize)], Data[int(len(Data) * trainingDataSize):]
90
91      # Weights [w₁, w₂, ..., w₀], w₀ being bias
92      Weights = [random.random() for w in range(len(Data[0].Attributes) + 1)]          # Generates a list of random weights
93
94      Loss = Training(TrainingData)
```

pav. 11 87-94 eilutės

96 eilutė išveda suapvalintus 2 vietos po kablelio svorius.
97-98 eilutės atitinkamai išveda tikslumą mokymo ir testavimo duomenims.
99 eilutė išveda suapvalintas 2 vietos po kablelio kiekvienos epochos paklaidas.
100 eilutė išveda paklaidą testavimo duomenims.

```
96      print("Svoriai: ", [round(w, 2) for w in [Weights[-1]] + Weights[:-1]])
97      print("Tikslumas mokymo duomenims:", Testing(TrainingData))
98      print("Tikslumas testavimo duomenims:", Testing(TestingData))
99      print("Kiekvienos epochos paklaidos:", [round(l, 2) for l in Loss])
100     print("Paklaida testavimo duomenims:", round(LossFunction(TestingData), 2))
```

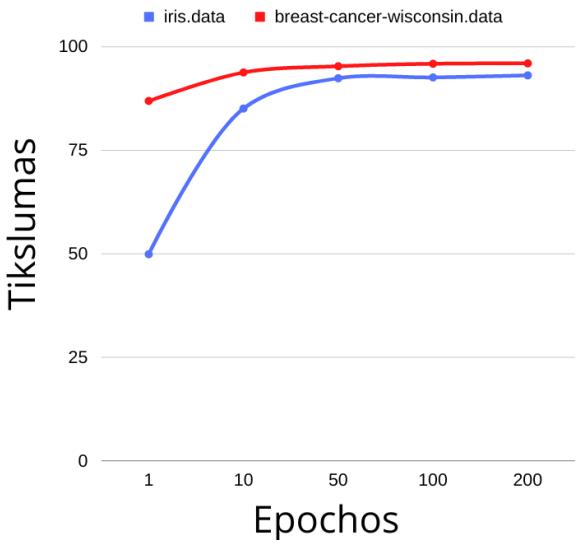
pav. 12 Programos išvedimas

Rezultatų priklausomybė nuo parametrų

Visiems tyrimams buvo imta 100 bandymų vidurkis.

Klasifikavimo tikslumo priklausomybė nuo epochų skaičiaus

Tyrime aktyvacijos funkcija yra sigmoidinė ir mokymosi greitis yra 0,1.



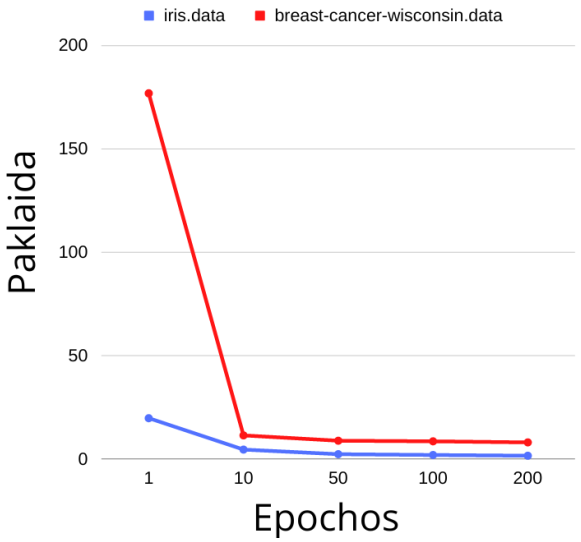
graf. 1 Klasifikavimo tikslumo priklausomybė nuo epochų skaičiaus

Duomenys, epochos	***iris.data***	***breast-cancer-wisconsin.data***
1	49,9 %	86,9 %
10	85,1 %	93,8 %
50	92,4 %	95,3 %
100	92,6 %	95,9 %
200	93,1 %	96,0 %

lent. 1 Klasifikavimo tikslumo priklausomybė nuo epochų skaičiaus

Paklaidos priklausomybė nuo epochų skaičiaus

Tyrime aktyvacijos funkcija yra sigmoidinė ir mokymosi greitis yra 0,1.



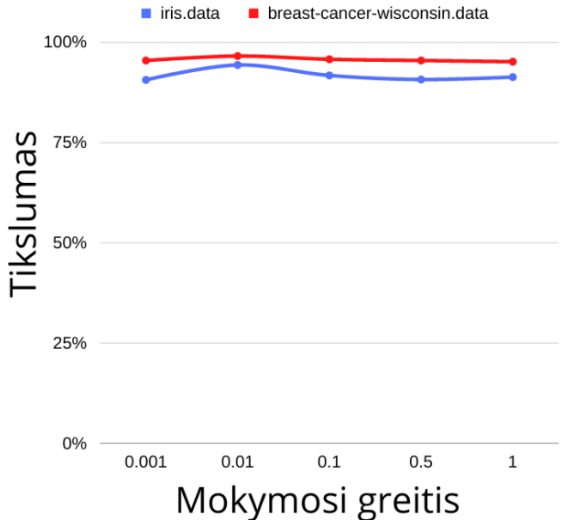
graf. 2 Paklaidos priklausomybė nuo epochų skaičiaus

Duomenys, epochos	***iris.data***	***breast-cancer-wisconsin.data***
1	49,9 %	86,9 %
10	85,1 %	93,8 %
50	92,4 %	95,3 %
100	92,6 %	95,9 %
200	93,1 %	96,0 %

lent. 2 Paklaidos priklausomybė nuo epochų skaičiaus

Rezultatų priklausomybė nuo mokymosi greičio

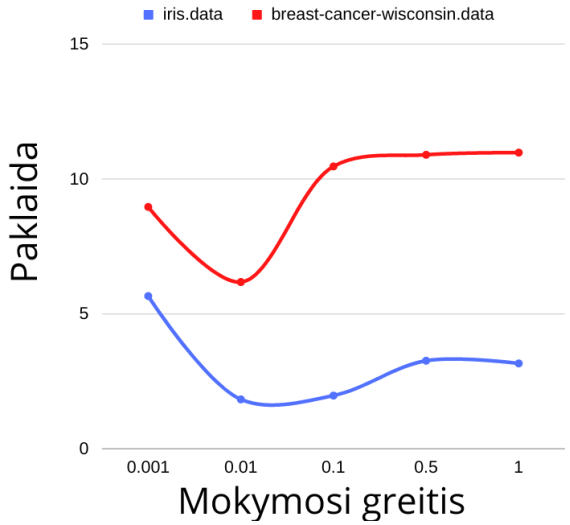
Tyrime aktyvacijos funkcijos yra sigmoidinės ir epochų skaičius yra 100.



graf. 3 Tikslumo priklausomybė nuo mokymosi greičio

Duomenys, mokymosi greitis	***iris.data***	***breast-cancer-wisconsin.data***
0,001	90,7 %	95,5 %
0,01	94,4 %	96,6 %
0,1	91,8 %	95,8 %
0,5	90,8 %	95,5 %
1	91,4 %	95,2 %

lent. 3 Tikslumo priklausomybė nuo mokymosi greičio



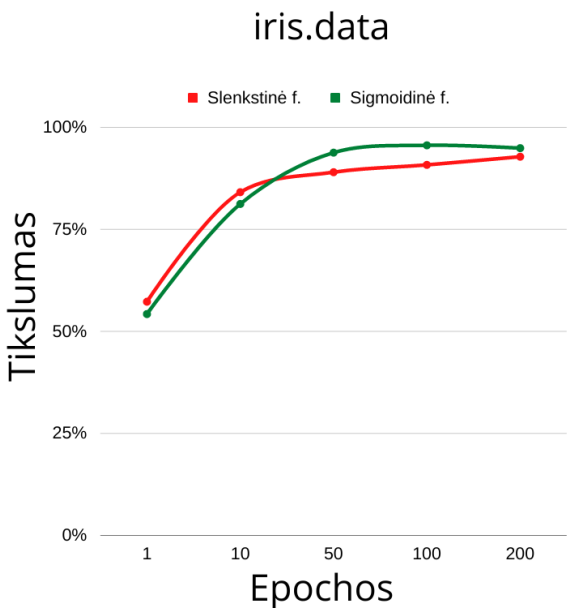
graf. 4 Paklaidos priklausomybė nuo mokymosi greičio

Duomenys, mokymosi greitis	***iris.data***	***breast-cancer-wisconsin.data***
0,001	5,67	8,97
0,01	1,85	6,19
0,1	1,99	10,47
0,5	3,28	10,90
1	3,18	10,98

lent. 4 Paklaidos priklausomybė nuo mokymosi greičio

Rezultatų priklausomybė nuo aktyvacijos funkcijos

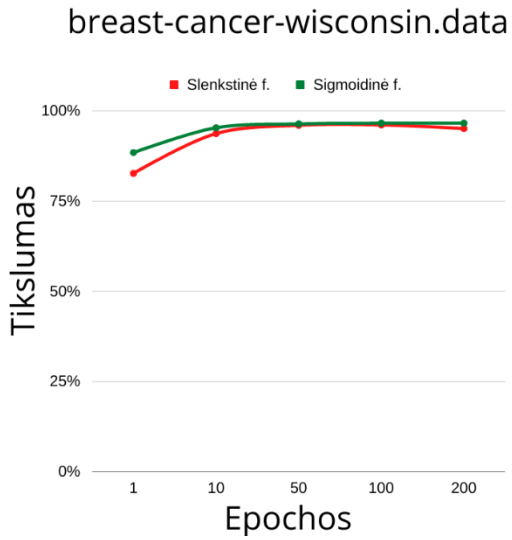
Tyrime mokymosi greitis yra 0,01.



graf. 5 Tikslumo priklausomybė nuo aktyvacijos funkcijos ir epochų skaičiaus (iris.data duomenys)

Funkcijos, epochos	***Slenkstinė***	***Sigmoidinė***
1	57,2 %	54,2 %
10	84,1 %	81,2 %
50	89,0 %	93,8 %
100	90,8 %	95,6 %
200	92,8 %	94,9 %

lent. 5 Tikslumo priklausomybė nuo aktyvacijos funkcijos ir epochų skaičiaus (iris.data duomenys)



graf. 6 Tikslumo priklausomybė nuo aktyvacijos funkcijos ir epochų skaičiaus (breast-cancer-wisconsin.data duomenys)

Funkcijos, epochos	***Slenkstinė***	***Sigmoidinė***
1	82,6 %	88,4 %
10	93,7 %	95,3 %
50	96,0 %	96,4 %
100	96,1 %	96,6 %
200	95,1%	96,6 %

lent. 6 Tikslumo priklausomybė nuo aktyvacijos funkcijos ir epochų skaičiaus (breast-cancer-wisconsin.data duomenys)

Tyrimų išvados

- Beveik visuose tyrimuose *breast-cancer-wisconsin.data* duomenų apmokytas neuronas pasirodė geriau. Turbūt didžiausią įtaką tam turėjo tai kad šie duomenys turėjo daugiau duomenų eilučių.
- Iš *graf. 1* galima pastebėti, kad nuo 50 epochos neurono tikslumo didėjimas praktiškai nustoja.
- Iš *graf. 2* galima pastebėti, kad paklaida labai greitai mažėja iki 10 epochos ir pastebimai sumažėja iki 50 epochos, bet nuo jos paklaidos mažėjimo greitis smarkiai sulėtėja.
- Iš *graf. 3* ir *graf. 4* galima pastebėti, kad geriausias mokymosi greitis yra 0,01 mūsų turimiems duomenims dėl didžiausio tikslumo ir mažiausios paklaidos.