**Podstawy Sztucznej Inteligencji – Laboratorium nr 5**

Wykonał: Kamil Wieniecki

Temat ćwiczenia: Budowa i działanie sieci Kohonena dla WTA.

1. Cel ćwiczenia

Celem ćwiczenia jest poznanie budowy i działania sieci Kohonena przy wykorzystaniu reguły WTA do odwzorowywania istotnych cech kwiatów.

1. Realizacja ćwiczenia

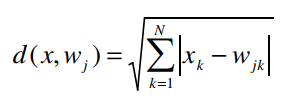
Wybrany przeze mnie język programowania do wykonania laboratorium to **Python.**

Sieć Kohonena nazywana jest inaczej mapą samoorganizującą. Bazuje na algorytmie uczenia bez nauczyciela, wagi na początku są losowane. Charakterystyczne jest tutaj uczenie konkurencyjne tzn. neurony konkurują ze sobą o prawo do reprezentacji danych wejściowych.

Metoda WTA (Winner Takes All – „Zwycięzca bierze wszystko”) oznacza, że tylko jeden element wyjściowy może znajdować się w stanie aktywnym, którego nazywamy zwycięzcą. Zwycięski neuron osiąga wartość minimalną wektora wag z wartością wektora wejściowego. Relacja ta wygląda następująco:



Wykorzystana przeze mnie miara odległości między wektorami to norma Manhattan, która przedstawia się następująco:



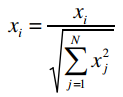
Dane wejściowe tworzą samoistną „bazę wiedzy” na podstawie której sieć uczy się i podejmuje decyzje. Ważna jest duża ilość rekordów, aby minimalizować ryzyko błędów.

Dane wejściowe zostały zaczerpnięte z linku podanego w instrukcji do laboratorium – pochodzą zatem z Wikipedii. Dane te zostały znormalizowane.

Normalizacja danych wejściowych polegała na tym samym co w poprzednim laboratorium – podziale każdej składowej wektora przez długość tego wektora. Przedstawia się następująco:



Każda składowa wektora opisana jest wzorem:



Potrzeba normalizacji dotyczyła tylko i wyłącznie danych wejściowych, wagi natomiast nie musiały już być normalizowane jako, że wektory uczące już takie były, więc wagi zachowywały się jako automatycznie znormalizowane.

Uczenie dotyczy tylko i wyłącznie zwycięskiego neuronu (wynika to z WTA), adaptacja wag opisana jest wzorem:



, gdzie **η – współczynnik uczenia**

**Xk – wartość składowej wektora wejściowego**

**Wj\*k – wartość składowej wektora wag**

Nadmiarowa ilość neuronów wynika z inicjalizacji wag losowo – przez co część neuronów może znaleźć się w strefie, gdzie nie ma danych lub ich liczba jest znikoma. Neurony te automatycznie zostają wykluczone z sieci, ponieważ nie mają szans na wygraną, nazywamy je neuronami martwymi.

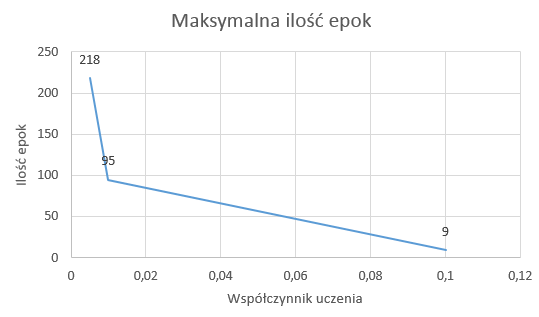
1. Wyniki

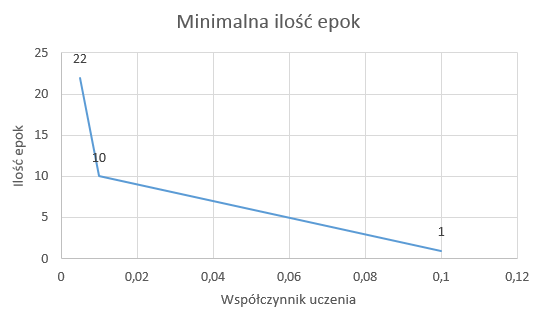
Liczba neuronów wybrana przeze mnie to 220, ilość kwiatów na wejście wynosiła po 20 każdego gatunku, natomiast testy czy sieć została nauczona została przeprowadzona na 8 rekordach testowych. Limit ustawiony przeze mnie to 10000 prób nauki.

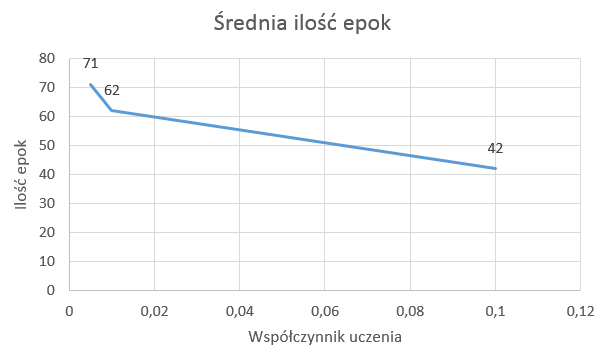
Poniższa tabela przedstawia zebrane wyniki dla poszczególnych współczynników uczenia.

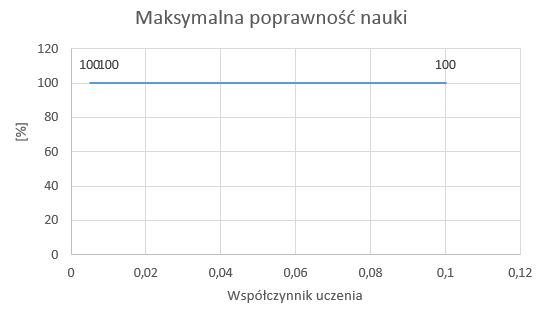
|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Współczynnik uczenia | 0.005 | | 0.01 | | 0.1 | |
| Lp. | Poprawność nauki [%] | Liczba epok | Poprawność nauki [%] | Liczba epok | Poprawność nauki [%] | Liczba epok |
| 1. | 100 | 137 | 91,66 | 11 | 91,66 | 7 |
| 2. | 100 | 32 | 95,83 | 95 | 95,83 | 7 |
| 3. | 100 | 61 | 100 | 47 | 95,83 | 2 |
| 4. | 95,83 | 48 | 100 | 10 | 91,66 | 9 |
| 5. | 95,83 | 22 | 100 | 27 | 95,83 | 9 |
| 6. | 100 | 37 | 100 | 56 | 100 | 3 |
| 7. | 100 | 218 | 100 | 30 | 95,83 | 3 |
| 8. | 100 | 18 | 95,83 | 62 | 95,83 | 1 |

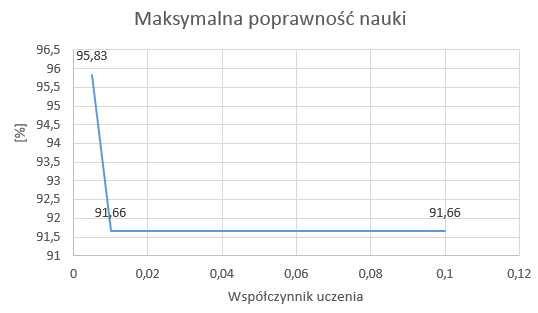
|  |  |  |  |
| --- | --- | --- | --- |
| Współczynnik uczenia | 0.005 | 0.01 | 0.1 |
| Maksymalna ilość epok | 218 | 95 | 9 |
| Minimalna ilość epok | 22 | 10 | 1 |
| Średnia ilość epok | 71,625 | 42,25 | 5,125 |
| Maksymalna poprawność nauki [%] | 100 | 100 | 100 |
| Minimalna poprawność nauki [%] | 95,83 | 91,66 | 91,66 |
| Średnia poprawność nauki [%] | 98,95 | 97,92 | 95,31 |

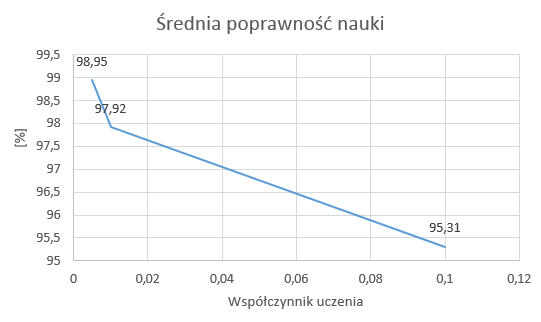












1. Analiza wyników

Jak wynika z powyższych wykresów zdecydowanie widać, że im mniejszy współczynnik nauki tym więcej epok do nauczenia się sieć potrzebowała. Występują dość spore różnice pomiędzy minimalną a maksymalną wartością ilości epok potrzebnych do nauki. Jednakże średnia ilość potrzebnych epok jest na całkiem zadowalającym poziomie. Średnia ilość dla poszczególnych współczynników nie różni się od siebie aż tak bardzo, jednak widać, że wraz ze wzrostem learning rate ilość epok stale spada.

Maksymalny % poprawności nauki w każdym przypadku wyniósł 100%, co jest wynikiem wręcz idealnym. Minimalna wartość wyniosła lekko powyżej 91%, co też jest bardzo dobrym wynikiem. Natomiast średnia poprawność nauki najlepiej wypadła przy jak najmniejszej wartości współczynnika nauki, co może sugerować, że lepiej uczyć sieć powoli i dłużej, ponieważ daje wtedy bardziej obiecujące i pewne wyniki niż w przypadku kiedy chcielibyśmy sieć nauczyć szybko.

1. Wnioski

Testowanie dla współczynnika uczenia większego niż 0,1 trwało bardzo długo – tj. sieć nie potrafiła poprawnie uczyć się wprowadzonych danych.

Sieć lepiej jest uczyć dłużej tzn. dobrać dość niski współczynnik uczenia, wtedy wyniki są pewniejsze. Odpowiednio dobrany współczynnik gwarantuje lepszą jakość nauki.

Sieć Kohonena bardzo dobrze poradziła sobie z zadanym problemem, gdzie jak widać poprawność wyników jest na bardzo wysokim poziomie. Bardzo ważną kwestią oczywiście jest też nasza baza danych – im więcej rekordów (informacji) posiadamy tym lepsza jakość nauczania, ponieważ sieć lepiej jest w stanie podejmować decyzje. Normalizacja danych wejściowych była potrzebna, ponieważ bez tego zawsze ten sam neuron wygrywałby „rywalizacje”, ponieważ jego wagi najszybciej zbliżyłyby się do wartości wejściowych nieznormalizowanych.

1. Listing kodu

**Normalize.py - - - - > służy do normalizacji danych wejściowych i zapisu do pliku**

from openpyxl import load\_workbook

from math import pow

from math import sqrt

### rekordy w excelu ###

### 1-50 ---> setosa

### 51-100 ---> versicolor

### 101-150 ---> virginica

workbook = load\_workbook('./Iris.xlsx', data\_only=True)

sheet = workbook.get\_sheet\_names()[0]

worksheet = workbook.get\_sheet\_by\_name(sheet)

text\_file = open("flowers.txt", "a")

def normalize(array, number):

sum = 0

for i in range(0, len(array)):

sum += pow(array[i], 2)

return number / sqrt(sum)

list = []

for row in worksheet.iter\_rows():

for i in range(0, len(row)):

list.append(row[i].value)

text\_file.write("[ ")

for i in range(0, 4):

result = normalize(list, list[i])

text\_file.write(str(result))

if i != 3:

text\_file.write(", ")

text\_file.write(" ],\n")

list.clear()

text\_file.close()

**Kohonen.py**

from random import uniform

class Kohonen:

def \_\_init\_\_(self, inputs):

"Konstruktor"

self.inputs = inputs

self.weights = []

for i in range(0, inputs):

self.weights.append(uniform(0, 1))

def learn(self, vector, learning\_rate):

"Metoda uczenia"

for i in range(0, self.inputs):

self.weights[i] += learning\_rate \* (vector[i] - self.weights[i])

**Data.py**

flower\_to\_learn = [

[

[ 0.8037727730153805, 0.5516087657948691, 0.2206435063179476, 0.03152050090256395 ],

[ 0.8281328733868766, 0.5070201265633938, 0.23660939239625042, 0.033801341770892925 ],

[ 0.8053330753805038, 0.5483118811101303, 0.22275170170099043, 0.03426949256938314 ],

[ 0.8000302474620596, 0.5391508189418229, 0.2608794285202369, 0.03478392380269825 ],

[ 0.7909649964604158, 0.5694947974514994, 0.22147019900891643, 0.03163859985841664 ],

[ 0.7841749862818121, 0.5663486012035309, 0.2468699030887186, 0.05808703602087497 ],

[ 0.7801093556963544, 0.5766025672538272, 0.23742458651628176, 0.05087669711063181 ],

[ 0.8021849185198158, 0.5454857445934748, 0.24065547555594474, 0.03208739674079263 ],

[ 0.8064236561517831, 0.5315065006454933, 0.2565893451392037, 0.03665562073417196 ],

[ 0.8180311900353678, 0.5175299365529878, 0.2504177112353167, 0.016694514082354447 ],

[ 0.8037351880735529, 0.550707443680027, 0.2232597744648758, 0.02976796992865011 ],

[ 0.7869910029407435, 0.55745196041636, 0.2623303343135812, 0.03279129178919765 ],

[ 0.8230721775566244, 0.5144201109728903, 0.24006271845401547, 0.01714733703242968 ],

[ 0.8025125990367244, 0.5598925109558542, 0.20529392068381325, 0.018663083698528475 ],

[ 0.8112086463534661, 0.5594542388644593, 0.16783627165933782, 0.02797271194322297 ],

[ 0.773811110254379, 0.5973278745823276, 0.20363450269852076, 0.054302534052938875 ],

[ 0.7942894410353187, 0.5736534851921746, 0.19121782839739154, 0.058836254891505096 ],

[ 0.8032741236882647, 0.5512665554723385, 0.22050662218893538, 0.04725141904048616 ],

[ 0.8068282029952023, 0.5378854686634681, 0.24063297282313048, 0.04246464226290538 ],

[ 0.7796488324493954, 0.5809148163348437, 0.22930848013217514, 0.04586169602643503 ],

[ 0.8173378965356997, 0.5146201570780331, 0.25731007853901655, 0.030271773945766654 ],

[ 0.7859185787017703, 0.5701762237640295, 0.23115252314757953, 0.061640672839354545 ],

[ 0.7757707459581176, 0.6071249316193964, 0.16864581433872122, 0.033729162867744245 ],

[ 0.8059779150822928, 0.5215151215238365, 0.2686593050274309, 0.07901744265512675 ],

[ 0.7761140001162655, 0.5497474167490214, 0.3072117917126884, 0.03233808333817773 ],

[ 0.8264745061341078, 0.49588470368046467, 0.2644718419629145, 0.033058980245364314 ],

[ 0.7977820578407535, 0.5424917993317123, 0.25529025850904113, 0.06382256462726028 ],

[ 0.806419649012008, 0.5427824560657746, 0.23262105259961768, 0.031016140346615696 ],

[ 0.8160942667466479, 0.5336000974881928, 0.21971768720102056, 0.03138824102871723 ],

[ 0.7952406381013439, 0.5414404344519789, 0.27072021722598943, 0.03384002715324868 ],

[ 0.8084658442238581, 0.5221341910612418, 0.2694886147412861, 0.03368607684266076 ],

[ 0.8222502813356185, 0.5177131401002042, 0.22840285592656068, 0.060907428247082854 ],

[ 0.7657831084892457, 0.6037905278472898, 0.22089897360266703, 0.014726598240177802 ],

[ 0.7786744728423064, 0.5946241428977613, 0.19820804763258706, 0.028315435376083872 ],

[ 0.8180311900353678, 0.5175299365529878, 0.2504177112353167, 0.016694514082354447 ],

[ 0.8251229524805619, 0.5280786895875597, 0.19802950859533486, 0.03300491809922248 ],

[ 0.8269975440259818, 0.5262711643801703, 0.19547214676977753, 0.03007263796458116 ],

[ 0.8180311900353678, 0.5175299365529878, 0.2504177112353167, 0.016694514082354447 ],

[ 0.8021241325140698, 0.5469028176232293, 0.23699122097006606, 0.03646018784154863 ],

[ 0.8077956848980377, 0.5385304565986918, 0.23758696614648167, 0.03167826215286423 ]

], ### setosa

[

[ 0.7670110293065677, 0.35063361339728816, 0.5149931196772669, 0.15340220586131353 ],

[ 0.7454975663530118, 0.3727487831765059, 0.5241779763419614, 0.17472599211398712 ],

[ 0.7551928518011777, 0.3392895421135726, 0.5362963730182276, 0.16417235908721253 ],

[ 0.7538491620232948, 0.31524601320974144, 0.5482539360169416, 0.17818252920550604 ],

[ 0.7581753965757455, 0.3265986323710904, 0.5365548960382199, 0.17496355305594127 ],

[ 0.7223296180302092, 0.3548285842955413, 0.5702602247606915, 0.1647418427086442 ],

[ 0.726348457433742, 0.38046823960815057, 0.5418790079267599, 0.1844694495069821 ],

[ 0.7591654715238996, 0.3718361493178283, 0.5112747053120139, 0.15493172888242845 ],

[ 0.7630185275970008, 0.3352657166714095, 0.5318007919615461, 0.15029152816304564 ],

[ 0.7246023348632823, 0.37623582771747355, 0.5434517511474617, 0.19508524400165292 ],

[ 0.7692307692307693, 0.3076923076923077, 0.5384615384615384, 0.15384615384615385 ],

[ 0.7392346162730675, 0.37588200827444107, 0.5262348115842176, 0.18794100413722054 ],

[ 0.7889275245573255, 0.2892734256710194, 0.5259516830382169, 0.13148792075955423 ],

[ 0.7308141200229885, 0.3474362209945355, 0.5630862891980404, 0.16772783082494816 ],

[ 0.7591170716092961, 0.3931141977976712, 0.48800383174883327, 0.1762236059093009 ],

[ 0.7694544446831323, 0.3560162355996582, 0.5053133666575794, 0.16078152575468435 ],

[ 0.706318918233044, 0.3783851347677022, 0.5675777021515532, 0.1891925673838511 ],

[ 0.7567649730125051, 0.3522871426092697, 0.5349545498881502, 0.13047671948491468 ],

[ 0.7644423782009608, 0.27125374710356676, 0.5548372099845683, 0.18494573666152278 ],

[ 0.7618518793947621, 0.34011244615837594, 0.5305754160070665, 0.14964947630968542 ],

[ 0.6985796007419847, 0.3788906309109069, 0.5683359463663603, 0.21312597988738513 ],

[ 0.770118538251249, 0.3534970339513929, 0.5049957627877042, 0.16412362290600388 ],

[ 0.7414330662236145, 0.29421947072365656, 0.5766701626183669, 0.17653168243419393 ],

[ 0.7365989486022555, 0.3381109928010353, 0.5675434522017379, 0.1449047112004437 ],

[ 0.7674169845534856, 0.34773582112579815, 0.5156082864968731, 0.15588157498742675 ],

[ 0.7678572552779229, 0.3490260251263286, 0.5119048368519487, 0.16287881172562 ],

[ 0.7646726945514761, 0.3148652271682549, 0.5397689608598655, 0.15743261358412744 ],

[ 0.7408857633640885, 0.3317398940436217, 0.5528998234060362, 0.1879859399580523 ],

[ 0.7335094873189085, 0.35452958553747244, 0.5501321154891814, 0.18337737182972713 ],

[ 0.7866747376502838, 0.3588340908580242, 0.48304589153964794, 0.13801311186847084 ],

[ 0.765218548535542, 0.3339135484518729, 0.5286964517154654, 0.15304370970710843 ],

[ 0.7724292477636958, 0.3370600353877945, 0.51963422122285, 0.14044168141158106 ],

[ 0.7643498123486223, 0.3558180160933242, 0.513959356579246, 0.15814134048592185 ],

[ 0.7077952502929945, 0.31850786263184755, 0.6016259627490452, 0.18874540007813187 ],

[ 0.6933340941942109, 0.3851856078856727, 0.5777784118285091, 0.19259280394283634 ],

[ 0.7152493550527451, 0.4053079678632222, 0.5364370162895589, 0.1907331613473987 ],

[ 0.7545734058794259, 0.3491309788397344, 0.5293276130795973, 0.16893434459987147 ],

[ 0.7753002085956333, 0.2830461078999931, 0.5414795107652043, 0.1599825827260831 ],

[ 0.7299244278720567, 0.39103094350288753, 0.5344089561206129, 0.1694467421845846 ],

[ 0.7471419369831229, 0.33960997135596493, 0.5433759541695439, 0.17659718510510178 ]

], ### versicolor

[

[ 0.6538774709871312, 0.3425072467075449, 0.6227404485591725, 0.25947518689965526 ],

[ 0.6905251239789963, 0.3214513508178087, 0.607185884878083, 0.22620650613105053 ],

[ 0.7149140499059414, 0.30207635911518654, 0.5940835062598668, 0.21145345138063057 ],

[ 0.6927679615927916, 0.31889318866969774, 0.6157937436380371, 0.1979337033122262 ],

[ 0.6861902181544978, 0.3167031776097682, 0.6122928100455518, 0.23224899691383002 ],

[ 0.7095370785652947, 0.28008042574945846, 0.6161769366488086, 0.1960562980246209 ],

[ 0.6705411755923674, 0.3421128446899833, 0.6158031204419699, 0.23263673438918864 ],

[ 0.7136655737001575, 0.2835109813329393, 0.6159031663439715, 0.17597233324113473 ],

[ 0.7141412519867053, 0.2664706164129497, 0.6182118300780434, 0.1918588438173238 ],

[ 0.6919878753680463, 0.34599393768402315, 0.5862675055201503, 0.2402735678361272 ],

[ 0.7156264473321344, 0.3523084048404354, 0.5614915202144439, 0.22019275302527214 ],

[ 0.7157654644679906, 0.3019635553224335, 0.5927432752625547, 0.21249287226393468 ],

[ 0.7171814811987383, 0.31640359464650214, 0.5800732568519207, 0.22148251625255153 ],

[ 0.692551795442857, 0.3037507874749373, 0.6075015749498746, 0.24300062997994984 ],

[ 0.6776792358909037, 0.3271554931887121, 0.5958903625937256, 0.28041899416175325 ],

[ 0.6958988736692162, 0.3479494368346081, 0.5762912547573196, 0.25008865772487454 ],

[ 0.706104739897313, 0.32589449533722137, 0.5974732414515725, 0.19553669720233283 ],

[ 0.6929909911756716, 0.3419955540866951, 0.6029921611528571, 0.1979974260501919 ],

[ 0.7060061789207179, 0.23839169677842423, 0.6326548876042797, 0.21088496253475988 ],

[ 0.727125847996813, 0.26661281093216477, 0.6059382066640108, 0.18178146199920325 ],

[ 0.7055893432004945, 0.327229840324867, 0.5828781530786694, 0.23519644773349813 ],

[ 0.6830792286072526, 0.3415396143036263, 0.597694325031346, 0.24395686735973307 ],

[ 0.7148654283077015, 0.25995106483916414, 0.6220257622937143, 0.1856793320279744 ],

[ 0.731224643079982, 0.3133819898914209, 0.5687302779510972, 0.2089213265942806 ],

[ 0.6959560108929095, 0.34278430387262704, 0.5920819794163558, 0.21813546610076268 ],

[ 0.7152945332140833, 0.3179086814284815, 0.5960787776784028, 0.17882363330352083 ],

[ 0.7278519543665409, 0.3287073342300507, 0.5634982872515155, 0.21131185771931832 ],

[ 0.7117121386170342, 0.35002236325427916, 0.5717031933153226, 0.2100134179525675 ],

[ 0.6959400158179216, 0.3044737569203407, 0.6089475138406814, 0.22835531769025555 ],

[ 0.7308985536964403, 0.30454106404018344, 0.588779390477688, 0.16242190082143118 ],

[ 0.7276615932660223, 0.2753314136682246, 0.5998291512057751, 0.18683203070343812 ],

[ 0.7157899883667153, 0.34430404503715417, 0.5798804969046808, 0.18121265528271271 ],

[ 0.6941774650196998, 0.30370264094611865, 0.6074052818922373, 0.2386235036005218 ],

[ 0.7236600467806032, 0.3216266874580459, 0.5858200378700121, 0.17230001113823887 ],

[ 0.6938541359029943, 0.29574110710619433, 0.63698084607488, 0.15924521151872 ],

[ 0.7315439854044734, 0.28501713717057403, 0.5795348455801672, 0.21851313849744008 ],

[ 0.6701748440536647, 0.3616816618702317, 0.5957109724921463, 0.2553047024966342 ],

[ 0.6980479904039343, 0.3381169953519057, 0.5998849917533811, 0.19632599730110653 ],

[ 0.7106690545187015, 0.3553345272593508, 0.5685352436149612, 0.21320071635561044 ],

[ 0.7241525806193215, 0.3253439130318691, 0.566728106571643, 0.22039426366675005 ]

] ### virginica

]

florwer\_to\_test = [

[

[ 0.8003330078112875, 0.5602331054679013, 0.20808658203093475, 0.04801998046867725 ],

[ 0.8609385732675535, 0.44003527078119403, 0.2487155878328488, 0.057395904884503565 ],

[ 0.7860903755415094, 0.5717020913029159, 0.2322539745918096, 0.035731380706432246 ],

[ 0.7888947910127371, 0.5522263537089159, 0.2524463331240759, 0.09466737492152845 ],

[ 0.7669389719546461, 0.571444724201501, 0.2857223621007505, 0.060152076231736955 ],

[ 0.8221058465189321, 0.5138161540743327, 0.23978087190135522, 0.05138161540743326 ],

[ 0.7772909266606612, 0.5791579453550025, 0.24385597699158001, 0.030481997123947502 ],

[ 0.7959478212368801, 0.5537028321647862, 0.24224498907209396, 0.03460642701029914 ],

[ 0.7983702482709741, 0.5573528148306801, 0.2259538438502757, 0.030127179180036764 ],

[ 0.8122836313809656, 0.5361071967114373, 0.22743941678667037, 0.03249134525523863 ]

], ### setosa

[

[ 0.7233711847707108, 0.3419572873461542, 0.5786969478165688, 0.15782644031360962 ],

[ 0.7326039145409681, 0.3602970071512958, 0.5524554109653201, 0.16813860333727135 ],

[ 0.7626299404054147, 0.34186859397484104, 0.5259516830382169, 0.1577855049114651 ],

[ 0.7698687947429833, 0.3541396455817723, 0.508113404530369, 0.15397375894859666 ],

[ 0.7354428354362763, 0.3545885099424904, 0.5515821265772073, 0.1707278010834213 ],

[ 0.7323961772671147, 0.38547167224584983, 0.5396603411441898, 0.15418866889833993 ],

[ 0.734460466367082, 0.3736728688534276, 0.5411813962704815, 0.16750852741705377 ],

[ 0.7572810334533846, 0.35421209629271216, 0.5252110393305731, 0.15878473282087097 ],

[ 0.7825805422808448, 0.3836179128827671, 0.4603414954593205, 0.16879188166841752 ],

[ 0.7431482021659327, 0.36505525720431775, 0.5345451980491795, 0.16948994084486185 ]

], ### versicolor

[

[ 0.6999703739013986, 0.32386688941706504, 0.5850498647534077, 0.250735656322889 ],

[ 0.7333788617780355, 0.32948905384230576, 0.5420626369663739, 0.24445962059267845 ],

[ 0.6905251239789963, 0.3214513508178087, 0.607185884878083, 0.22620650613105053 ],

[ 0.6919350210150538, 0.3256164804776724, 0.6003553858807085, 0.234036845343327 ],

[ 0.6891487078995805, 0.3394314531445695, 0.5862906917951656, 0.2571450402610375 ],

[ 0.721557247888883, 0.32308533487561925, 0.5600145804510733, 0.24769875673797473 ],

[ 0.7296535932873784, 0.28954507670134066, 0.5790901534026813, 0.2200542582930189 ],

[ 0.7165389871201644, 0.3307103017477682, 0.5732311896961315, 0.22047353449851215 ],

[ 0.6746707198515326, 0.36998071733793725, 0.5876164334190769, 0.25028107349331047 ],

[ 0.6902591585738451, 0.35097923317314156, 0.5966646963943406, 0.21058753990388496 ]

] ### virginica

]

**Main.py**

from Kohonen import Kohonen

from math import sqrt

from Data import flower\_to\_learn

from Data import florwer\_to\_test

def learn(kohonens):

"Funkcja uczenia"

eras = 0

winners = []

for i in range(0, FLOWERS):

winners.append([])

for j in range(0, LEARN\_SAMPLES):

winners[i].append(-1)

while unique(winners) != True:

for i in range(0, FLOWERS):

for j in range(0, LEARN\_SAMPLES):

winner = get\_winner(kohonens, flower\_to\_learn[i][j])

kohonens[winner].learn(flower\_to\_learn[i][j], LEARNING\_RATE)

for i in range(0, FLOWERS):

for j in range(0, LEARN\_SAMPLES):

winners[i][j] = get\_winner(kohonens, flower\_to\_learn[i][j])

eras += 1

if eras == LIMIT:

break

return eras

def unique(winners):

"Funkcja sprawdza czy siec juz sie nauczyla rozpoznawac kwiaty"

"Petla sprawdza czy kazdy typ ma jednego zwyciezce"

for i in range(0, FLOWERS):

for j in range(1, LEARN\_SAMPLES):

if winners[i][0] != winners[i][j]:

return False

"Petla sprawdzajaca wszystkie gatunki, czy jest rozny od zwyciezcow pozostalych gatunkow"

for i in range(0, FLOWERS):

for j in range(0, FLOWERS):

if i != j:

if winners[i][0] == winners[j][0]:

return False

return True

def get\_winner(kohonens, vector):

"Zwraca wygranego"

winner = 0

minimum\_distance = vector\_distance(kohonens[0].weights, vector)

for i in range(1, NEURONS):

current\_distance = vector\_distance(kohonens[i].weights, vector)

if current\_distance < minimum\_distance:

winner = i

minimum\_distance = current\_distance

return winner

def vector\_distance(vector, vector2):

"Zwraca odleglosc pomiedzy dwoma wektorami (Manhattan distance)"

sum = 0

for i in range(0, len(vector)):

sum += abs(vector[i] - vector2[i])

return sqrt(sum)

### Dane wejściowe

LEARNING\_RATE = 0.01

INPUTS = 4

NEURONS = 220

FLOWERS = 3

LEARN\_SAMPLES = 20

TEST\_SAMPLES = 8

LIMIT = 10000

success = 0

fail = 0

while success != 10 and fail != 100:

kohonens = []

for i in range(0, NEURONS):

kohonens.append(Kohonen(INPUTS))

eras = learn(kohonens)

if eras != LIMIT:

success += 1

print("Po nauce:\n")

for i in range(0, FLOWERS):

winner = get\_winner(kohonens, flower\_to\_learn[i][0])

print("Kwiat [", i, "] wygral =", winner)

print("\nPo testowaniu:")

for i in range(0, FLOWERS):

print("\n")

for j in range(0, TEST\_SAMPLES):

winner = get\_winner(kohonens, florwer\_to\_test[i][j])

print("Kwiat [", i, "][", j, "]", "winner =", winner)

print("Ilosc epok:", eras)

else:

fail += 1

print("Ilosc niepowodzen w nauce:", fail)

1. Bibliografia

<http://aragorn.pb.bialystok.pl/~gkret/SSN/SSN_w10.PDF>

<https://pl.wikipedia.org/wiki/Sie%C4%87_Kohonena>

<http://galaxy.agh.edu.pl/~vlsi/AI/koho_t/>

https://en.wikipedia.org/wiki/Iris\_flower\_data\_set