МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РЕСПУБЛИКИ КАЗАХСТАН

СӘТБАЕВ УНИВЕРСИТЕТІ



**Кафедра программной инженерии**

**Дисциплина:** BNN

**Финальный экзамен**

**Экзаменационный билет. Вариант 4**

Время занятий: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

ФИО: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Экзамен состоит из трех частей

1. Тестовые вопросы (20 баллов / 50 %/30')   
2. Открытые вопросы (10 баллов / 25 %/25' )

3. Задача (10 баллов / 25 %/25')   
**Часть 1. Ответьте на вопросы теста , выбрав подходящий вариант (смотреть ниже)**

**Часть 2. Вопрос**

1. Полиномиальная регрессия. Математические основы, реализация. Типы задач, которые могут быть решены с помощью полиномиальной регрессии.
2. Искусственные нейронные сети. Многослойная нейронная сеть. Обучение многослойных нейронных сетей.

**Часть 3. Задача X1**

В данной реализации нейронной сети необходимо заменить активационную функцию нейронной сети на Tanh().

1. Пронумеруйте строки программы, в которых нужно внести изменения.
2. Ниже напишите примерный код, который следует записать в этих строках

*# Imports*

**import** **numpy** **as** **np**

**import** **matplotlib.pyplot** **as** **plt**

*# Each row is a training example, each column is a feature [X1, X2, X3]*

X=np.array(([0,0,1],[0,1,1],[1,0,1],[1,1,1]), dtype=float)

y=np.array(([0],[1],[1],[0]), dtype=float)

*# Class definition*

**class** **NeuralNetwork**:

**def** \_\_init\_\_(self, x,y,alpha):

self.input = x

self.weights1= np.random.rand(self.input.shape[1],4) *#\*0.001 # considering we have 4 nodes in the hidden layer*

self.weights2 = np.random.rand(4,1) *#\*0.001*

self.y = y

self.alpha=alpha

self.output = np. zeros(y.shape)

*# Activation function*

**def** sigmoid(self,t):

**return** 1/(1+np.exp(-t))

**def** sigmoid\_derivative(self,p):

**return** p \* (1 - p)

**def** feedforward(self):

self.layer1 = self.sigmoid(np.dot(self.input, self.weights1))

self.layer2 = self.sigmoid(np.dot(self.layer1, self.weights2))

**return** self.layer2

**def** backprop(self):

alpha=0.1

d\_weights2 = np.dot(self.layer1.T, 2\*(self.y -self.output)\*self.alpha\*self.sigmoid\_derivative(self.output))

d\_weights1 = np.dot(self.input.T, np.dot(2\*(self.y -self.output)\*self.alpha\*self.sigmoid\_derivative(self.output), self.weights2.T)\*self.sigmoid\_derivative(self.layer1))

self.weights1 += d\_weights1

self.weights2 += d\_weights2

**def** train(self, X, y):

self.output = self.feedforward()

self.backprop()

arrLoss=[]

iterat=[]

nn=NeuralNetwork(X,y,alpha=0.07)

**for** i **in** range(3500): *# trains the NN 1,000 times*

L=np.mean(np.square(y - nn.feedforward()))

arrLoss.append(L)

iterat.append(i)

**if** i % 1000 ==0:

print ("for iteration # " + str(i) + "**\n**")

print ("Input : **\n**" + str(X))

print ("Actual Output: **\n**" + str(y))

print ("Predicted Output: **\n**" + str(nn.feedforward()))

print ("Loss: **\n**" + str(np.mean(np.square(y - nn.feedforward())))) *# mean sum squared loss*

print ("**\n**")

nn.train(X, y)

Тестовые вопросы

<https://docs.google.com/forms/d/e/1FAIpQLSdM7hnJZiFK3kMj5Bgwnw9ntf4BCukqnnIF3KnqmC1nsVtfTg/viewform?usp=sf_link>