**13-я проблема Гильберта.** Можно ли решить общее уравнение седьмой степени с помощью функций, зависящих только от двух переменных? (Можно ли функцию нескольких переменных представитьт в виде суперпозиции нескольких непрерывных функций двух переменных?)

**Теорема Колмогорова — Арнольда.** Многомерная непрерывная функция может быть представлена в виде конечной композиции непрерывных функций одной переменной и операции сложения.

Или: любая непрерывная функция, n вещественных переменных, может быть представлена в виде суммы функций, имеющих своим аргументом суммы непрерывных функций одного аргумента.

$$f\left(x\_{1}, …, x\_{n}\right)=\sum\_{k=0}^{2n}Φ\_{k}\left(\sum\_{p=1}^{n}φ\_{k,p}\left(x\_{p}\right)\right)$$

**Теорема Цибенко.** Пусть $σ(x)$ непрерывная сигмоидная функция

$$σ\left(x\right)=\left\{\begin{array}{c}1 при x\rightarrow \infty \\0 при x\rightarrow -\infty \end{array}\right.$$

тогда конечная сумма

$$G\left(x\right)=\sum\_{i}^{}α\_{i}σ\left(w\_{i}∙x+b\_{i}\right)$$

сколь угодно точно приближает любую непрерывную функцию $f(x)$. Другими словами, для сколь угодно малого $ε$ выполняется условие

$$\left|f\left(x\right)-G(x)\right|<ε для ∀x$$

 Проложим тропинку к нейронным сетям. Для начала в качестве сигмоидной функции используем функцию индикатор

$$I\left(x\right)=\left\{\begin{array}{c}1 если x "истина"\\0 если x "ложь" \end{array}\right.$$

тогда $σ(x)≈I\left(x>0\right)$

Нейрон:

$$y=I\left(\sum\_{i=1}^{N}w\_{i}x\_{i}>h\right)$$

или

$$y=I\left(\sum\_{i=1}^{N}w\_{i}x\_{i}+b>0\right)$$

здесь $x\_{i}$ – входные сигналы;

$w\_{i}$ –набор весов;

*b* – смещение или *h=-b* – порог срабатывания;



 Что можно получить в простейшем случае одного нейрона с двумя входами?

$$y=I\left(w\_{1}x\_{1}+w\_{2}x\_{2}+b>0\right)$$

$$w\_{1}=1, w\_{2}=1, b=0 :y=x\_{1} or x\_{2} $$

$$w\_{1}=1, w\_{2}=1, b=-1 :y=x\_{1} and x\_{2} $$

Для операции $xor$ одного нейрона недостаточно. Применим формулу:

$$x\_{1} xor x\_{2}= \left(x\_{1} or x\_{2}\right) and ((not x\_{1}) or (not x\_{2}))$$

Используем

$$w\_{1}=-1, w\_{2}=-1, b=2 :y=(not x\_{1}) or (not x\_{2}) $$

Тогда

$$y= I\left(\left(I\left(x\_{1}+x\_{2}>0\right)\right)+\left(I\left(-x\_{1}-x\_{2}>-2\right)\right)>1\right) : y=x\_{1} xor x\_{2}$$

От одного нейрона мы перешли к суперпозиции нейронов: выход одних нейронов подаём на вход других.

Для индикатора "столбик"

$$I\left(x\_{i}\leq x<x\_{i+1}\right)= I\left(x-x\_{i}\geq 0\right)-I\left(x-x\_{i+1}\geq 0\right) x\_{i}<x\_{i+1} $$

комбинация нейронов имеет вид:



 Если взять скрытый слой с достаточно большим числом нейронов, то мы можем аппроксимировать любую гладкую функцию.



$$f\left(x\right)=\sum\_{i=0}^{N-1}f\left(\frac{x\_{i}+x\_{i+1}}{2}\right)∙I\left(x\_{i}\leq x<x\_{i+1}\right)$$

Выразим индикатор $I\left(x\_{i}\leq x<x\_{i+1}\right)$ через нейроны:

$$I\left(x\_{i}\leq x<x\_{i+1}\right)=I\left(x-x\_{i}\geq 0\right)-I\left(x-x\_{i+1}\geq 0\right)$$

$$f\left(x\right)=\sum\_{i=0}^{N-1}f\left(\frac{x\_{i}+x\_{i+1}}{2}\right)∙\left(I\left(x-x\_{i}\geq 0\right)-I\left(x-x\_{i+1}\geq 0\right)\right)$$

$$f\left(x\right)=\sum\_{i=0}^{N-1}f\left(\frac{x\_{i}+x\_{i+1}}{2}\right)∙I\left(x-x\_{i}\geq 0\right)-\sum\_{i=0}^{N-1}f\left(\frac{x\_{i}+x\_{i+1}}{2}\right)∙I\left(x-x\_{i+1}\geq 0\right)$$



здесь

$$b=0 w\_{i}=\pm f\left(\frac{x\_{i}+x\_{i+1}}{2}\right)$$

(здесь *+f* и *-f* для двух рядом стоящих весов)

***Логистическая регрессия.***

Основная идея логистической регрессии заключается в том, что пространство исходных значений может быть разделено линейной границей (т. е. прямой) на две соответствующих классам области.

Уравнение

$$\sum\_{i=1}^{n}w\_{i}x\_{i}=u$$

задаёт (n-1)-гиперплоскость в n-мерном пространстве. Пространство разбивается на две области

$$\sum\_{i=1}^{n}w\_{i}x\_{i}>u и \sum\_{i=1}^{n}w\_{i}x\_{i}<u$$

соответствующие разделению множества объектов на два класса (линейный классификатор).

 Этот классификатор можно реализовать через один нейрон. Ограничимся двумерным случаем:

$$I\left(w\_{0}x+w\_{1}y+b>0\right)$$

Значение 0 или 1 на выходе соответствуют принадлежности к одному из двух кластеров.

 Теперь относительно значений весов $w\_{0} w\_{1}$ и смещения *b*.

 Если известны центры двух кластеров $\left(X\_{1},Y\_{1}\right) и \left(X\_{2},Y\_{2}\right)$, то прямая проходящая через эти центры имеет вид:

$$\frac{\left(x-X\_{1}\right)}{(X\_{2}-X\_{1})}=\frac{\left(y-Y\_{1}\right)}{(Y\_{2}-Y\_{1})} или y=ax+b, где a=\frac{\left(Y\_{2}-Y\_{1}\right)}{(X\_{2}-X\_{1})}$$

Координаты центр соответствующего отрезка равны

$$X\_{0}=\frac{X\_{1}+X\_{2}}{2} Y\_{0}=\frac{Y\_{1}+Y\_{2}}{2}$$

Уравнение прямой, проходящей через точку $\left(X\_{0},Y\_{0}\right)$ и перпендикулярной прямой проходящей через центры кластеров имеет вид:

$$\frac{\left(y-Y\_{0}\right)}{\left(x-X\_{0}\right)}=-\frac{1}{a} или x+ay-X\_{0}-aY\_{0}=0 $$

Этому уранению следует придать симметричный относительно x-y вид, в этом случае мы избежим неприятностей при $X\_{2}=X\_{1}$.

$$\left(X\_{2}-X\_{1}\right)x+\left(Y\_{2}-Y\_{1}\right)y-\left(X\_{2}-X\_{1}\right)X\_{0}-\left(Y\_{2}-Y\_{1}\right)Y\_{0}=0$$



 Если кластеры одинаковы по размеру, то эта прямая будет являться разделителем в линейном классификаторе. В этом случае для весов $w\_{0} w\_{1}$ и смещения *b* получаем следующие значения:

$$w\_{0}=X\_{2}-X\_{1}, w\_{1}=Y\_{2}-Y\_{1}, b=-\left(X\_{2}-X\_{1}\right)X\_{0}-\left(Y\_{2}-Y\_{1}\right)Y\_{0}$$

 ***Программная реализация:***

# Linear classifier

import numpy as np

import matplotlib.pyplot as plt

import pandas as pd

from sklearn import datasets

from sklearn import preprocessing

from math import exp

POINT\_N = 300

DIM\_N = 2

CLUST\_N = 2

data, target, Center = datasets.make\_blobs(n\_samples=POINT\_N, centers=CLUST\_N, cluster\_std=0.8, n\_features=DIM\_N, center\_box=(-10,10), random\_state=0, return\_centers=True)

slope = (Center[1][1]-Center[0][1])/(Center[1][0]-Center[0][0])

C0 = [(Center[0][0]+Center[1][0])/2, (Center[0][1]+Center[1][1])/2]

print("w1=", Center[1][0]-Center[0][0])

print("w2=", Center[1][1]-Center[0][1])

print("b=", -(Center[1][0]-Center[0][0])\*C0[0] - (Center[1][1]-Center[0][1])\*C0[1])

plt.figure(figsize=(7,9))

plt.scatter(data[:,0], data[:,1], c='black', marker='o')

plt.scatter(Center[:,0], Center[:,1], c='red', marker="\*", s=100)

plt.axline((Center[0][0], Center[0][1]), slope=slope, color="blue", linestyle=(0, (5, 5)))

plt.axline((C0[0], C0[1]), slope=-1.0/slope, color="blue", linestyle=(0, (5, 5)))

plt.show()



class NNET0:

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

 self.input\_nodes = DIM\_N

 self.output\_nodes = 1

 self.weights\_input\_to\_output = np.array([[Center[1][0]-Center[0][0]], [Center[1][1]-Center[0][1]]])

 self.output\_bias = np.array([ -(Center[1][0]-Center[0][0])\*C0[0] - (Center[1][1]-Center[0][1])\*C0[1] ])

 def activation\_function(self, x):

 return (1 if x>0 else 0)

 def run(self, features):

 input\_output = np.dot(features, self.weights\_input\_to\_output)

 return self.activation\_function(input\_output+self.output\_bias)

network = NNET0()

pred = []

for i in range(POINT\_N):

 pred.append(network.run(data[i]))

plt.figure(figsize=(8,8))

for i in range(POINT\_N):

 if pred[i]:

 plt.scatter(data[i][0], data[i][1], c='blue', marker='o')

 else:

 plt.scatter(data[i][0], data[i][1], c='green', marker='o')

plt.scatter(Center[:,0], Center[:,1], c='red', marker="\*", s=100)

plt.axline((C0[0], C0[1]), slope=-1.0/slope, color="black", linestyle=(0, (5, 5)))

plt.show()

print(1.0\*sum(target==pred)/POINT\_N)

print('----------------------------------------------------------')





**Задание:**



 Построить нейросеть для реализации линейного классификатора в соответствии с приведённым рисунком.