Laboratorinis darbas Nr 2.

Vienasluoksnis perceptronas (Python)
Klasifikavimo uždavinys

Bendrieji nurodymai

Laboratorinio darbo ataskaitą pateikite raštu. Žodžiu pristatykite sukurtas programas, pademonstruokite jų veikimą, grafinius rezultatus.

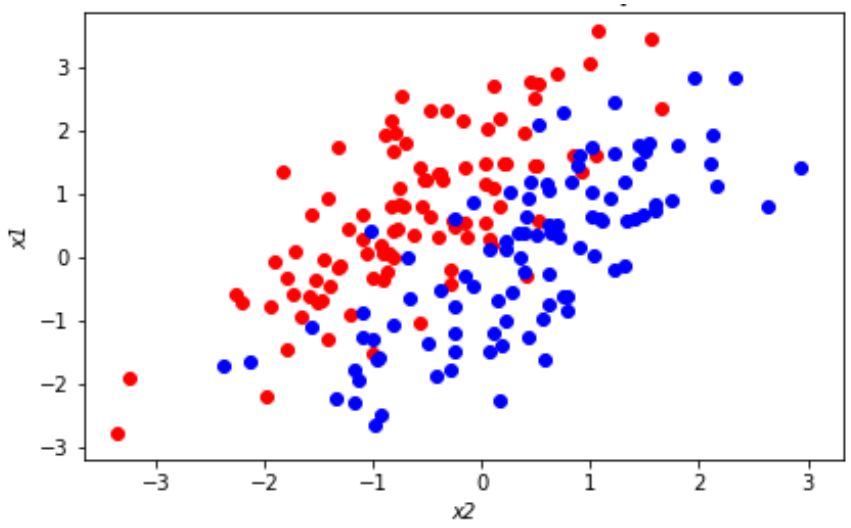
Naudokite Python programą *single_layer_perceptron.ipynb*

1. Tikslai
Išmokti panaudoti vienasluoksnį perceptroną klasifikavimo uždaviniui spręsti.

2. Vienasluoksnis perceptronas kaip klasifikatorius.

2.1 Duomenys

Programoje sukuriami dvimačiai Gauso skirstinio duomenys, kurie suskirstomi į dvi klases. Pirmoji klasė pažymėta raudonai, antroji – mėlynai.



Mūsų tikslas – išmokyti vienasluoksnį perceptroną atskirti šias dvi klases išvedant tiesinį skiriamąjį paviršių tarp jų.

Kiekvienos klasės duomenų struktūrą apsprendžia vidurkio vektoriai `mean1, mean2` ir kovariacinės matricos `cov1, cov2`. Programoje naudojami dydžiai, kuriuos galite keisti:

```
mean1 = np.array([-0.5, 1])  #  two-dimensional mean vector [x1 x2] of
class 1
mean2 = np.array([0.5, 0])   #  two-dimensional mean vector [x1 x2] of
class 2

var1=1 # variance of x1 feature (the same for both classes)
var2=2 # variance of x2 feature (the same for both classes)
cor12=0.8 #correlation coefficient between x1 and x2 (the same for both
classes)
```

Mokymo ir testavimo imtys (lygios abiems klasėms):

```
N_train=100 # number of training samples
N_test=100 # number of testing samples
```

Mokymo imtys žymimos: `data_train1, data_train2.`
Tesstavimo imtys žymimo: `data_test1, data_test2.`

Targetai (norimi išėjimai):
Mokymo imtis, 1-a klasė: `targets_train1` – nuliai
Mokymo imtis, 2-a klasė: `targets_train2` – vienetai
Testinė imtis, 1-a klasė: `targets_test1` – nuliai
Testinė imtis, 2-a klasė: `targets_test2` – vienetai

2.2. Vienasluoksnio perceptrono mokymas ir testavimas

Mokymo parametrai:

```
#parameters for perceptron training
niu = 0.2 # learning Rate
epochs = 200 #number of iterations
```

Perceptrono mokymui ir testavimui bus naudojamos funkcijos:

```
perceptron = Perceptron(data_train, targets_train)

#training
mse=perceptron.train(epochs, niu)

#training errors
error_train, error_train_percent]=perceptron.errors(data_train, targets_train)

#testing errors
[error_test, error_test_percent]=perceptron.errors(data_test, targets_test)
```

Mokymas atliekamas metode:

```
def train(self, its, niu):
```

Išanalizuokite programą. Ypatingą dėmesį skirkite mokymosi taisyklei suprasti (pažymėt raudonai):

```
def train(self, its, niu):
...

for i in range(N): #over the samples
     cost_prime = 2*(activation[i] - self.targets[i])
     #weight change for a given sample
     for j in range(p+1): #over the features
     delta_weights[j][i] = cost_prime * inputs1[i][j] *
                                    self.sigmoid_deriv(z[i])

     #average of the weight change over the learning samples and transpose
     delta_avg = np.array([np.average(delta_weights, axis=1)]).T

     #weight update
     self.weights = self.weights - niu * delta_avg
```

Programos pabaigoje yra nubraižomas vidutinės kvadratinės paklaidos *mse* kitimas mokymosi metu, mokymo duomenys, suformuotas skiriamasis paviršius, testavimo duomenys, suformuotas skiriamasis paviršius, atspausdinami klasifikavimo klaidų įverčiai.

Užduotis Nr. 1. Ištirkite, kaip klasifikavimo tikslumą įtakoja mokymo iteracijų skaičius. Kaip kinta vidutinė kvadratinė paklaida mokymosi metu, ką tai reiškia?
Pastaba: kiekvieną kartą bus generuojami nauji duomenys. Norėdami dirbti su tais pačiais duomenimis, parašykite papildomą procedūrą duomenims išsaugoti.
Naudojant tuos pačius duomenis, galėsite korektiškai palyginti rezultatus.

Užduotis Nr. 2. Ištirkite, kaip klasifikavimo tikslumą įtakoja mokymosi greitis niu (`niu`>0).

Užduotis Nr. 3. Ištirkite, kaip klasifikavimo tikslumą įtakoja mokymo imties dydis `N_train`.

Užduotis Nr. 4. Ištirkite, kaip klasifikavimo tikslumą įtakoja atstumas tarp klasių, kuris išreiškiamas Mahalanobio atstumu (*Mahalanobis distance*).
Mahalanobio atstumas didėja didėjant skirtumams tarp vidurkių, atsižvelgiant į klasių kovariacines matricas. Kuo Mahalanobio atstumas didesnis, tuo geriau klasės atsiskiria.

Pakeiskite klasių vidurkius `mean1, mean2`; sugeneruokite duomenis iš naujo.
Galite pakeisti požymių kovariacijas `var1, var2,` koreliaciją `cor12` , suformuoti skirtingas klasių kovariacines matricas `cov1, cov2`.

```
mean1 = np.array([-0.5, 1])  #  two-dimensional mean vector [x1 x2] of
class 1
```

```
mean2 = np.array([0.5, 0])   #  two-dimensional mean vector [x1 x2] of
class 2
var1=1 # variance of x1 feature (the same for both classes)
var2=2 # variance of x2 feature (the same for both classes)
cor12=0.8 #correlation coefficient between x1 and x2 (the same for both
classes)
```

Užduotis Nr. 5 (tyrėjams). Pasirinkite ir suklasifikuote savo duomenis.

Ataskaitoje pateikite trumpą darbo rezultatų aprašymą. Turėkite programos kodą, kurį būtina suprasti.