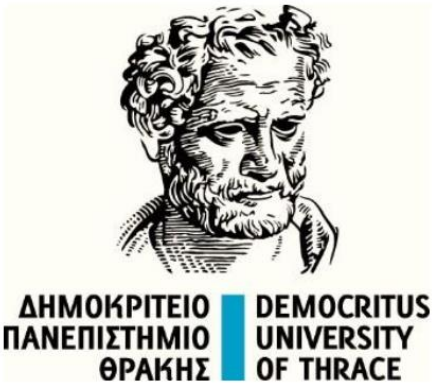


1Η ΕΡΓΑΣΙΑ

Στο μάθημα:
«Αναγνώριση Προτύπων»
Καθηγητής: Νικόλαος Μητιανούδης

Στέλιος Μούσλεχ
ΑΜ:57382
20/10/2020 ,Ξάνθη

Εισαγωγή

Η υλοποίηση του κώδικα των ερωτημάτων 2,3,4 έγινε σε **Python 3** . Για τον τύπο δεδομένων μαθηματικού πίνακα αλλά και μαθηματικών/στατιστικών συναρτήσεων έγινε χρήση των βιβλιοθηκών:

NumPy

SciPy

(Η βιβλιοθήκη αυτή περιέχει και χτίζει πάνω στην βιβλιοθήκη **NumPy**. Για καλύτερη πρακτική κώδικα όσες συναρτήσεις προϋπάρχουν στην NumPy θα καλούνται από τη NumPy και όχι από τη SciPy αν και πρακτικά είναι το ίδιο.
Π.χ:

X=numpy.array([0,1])

X=scipy.array([0,1])

Και τα δύο δημιουργούν ένα numpy array αλλά στο κώδικα μου θα χρησιμοποιηθεί ο πρώτος τρόπος).

Τα αρχεία κώδικα που είναι μαζί με το report οργανώνονται ως εξής:

Ex1.py : Ο κώδικας για το ερώτημα 2

Ex1-2Dice.py : Ο κώδικας για το ερώτημα 2

Ex1-GradientDescent.py : Ο κώδικας για το ερώτημα 4 (Μόνο Gradient Descent)

Ex1-Newton.py : Ο κώδικας για το ερώτημα 4 (Newton Method)

1ο ΕΡΩΤΗΜΑ

Α) Από την θεωρία γνωρίζουμε ότι για διακριτές τυχαίες μεταβλητές, όπως έχουμε στην περίπτωση του ζαριού ,η συνάρτηση πυκνότητας πιθανότητας ορίζεται γενικά ως :

$$f_X(x) = \sum_k P_X[x_k]\,\delta(x - x_k)$$

Θεωρώντας ότι έχουμε ένα απλό ζάρι 6 πλευρών ο δειγματοχώρος στην περίπτωση αυτή είναι :

$$\Omega = \{1, 2, 3, 4, 5, 6\}$$

Και τα στοιχεία του έχουν ίσες πιθανότητες στην ρίψη του ζαριού (έχουμε μια ομοιόμορφη διακριτή κατανομή)

Με βάση τα παραπάνω, η πιθανότητα του αποτελέσματος της ρίψης να είναι z1 ισούται με (αφού το κάθε αποτέλεσμα είναι ισοπίθανο):

$$P(Z = z1) = \frac{1}{6}$$ όπου z1=1,2,3,4,5,6

Άρα η συνάρτηση πυκνότητας πιθανότητάς του είναι:

$$f(z_1) = p(Z = z_1) = \frac{1}{6}$$

Β) Για διακριτές τυχαίες μεταβλητές η μέση τιμή ή αναμενόμενη τιμή ορίζεται ως

$$E[X] = \sum_k x_k P_X(x_k)$$

Άρα στην περίπτωση μας:

$$E[Z] = 1 \cdot \frac{1}{6} + 2 \cdot \frac{1}{6} + 3 \cdot \frac{1}{6} + 4 \cdot \frac{1}{6} + 5 \cdot \frac{1}{6} + 6 \cdot \frac{1}{6} = 3.5$$

Η μεταβλητότητα ορίζεται ως:

$$Var[X] = \sigma^2 = E\left[\left(X - E[X]\right)^2\right]$$

Και μπορούμε να αποδείξουμε ότι:

$$\begin{aligned}\mathrm{Var}[X] &= \mathrm{E}\left[(X-\mathrm{E}[X])^2\right] \\ &= \mathrm{E}\left[X^2+\mathrm{E}[X]^2-2\mathrm{E}[X]X\right] \\ &= \mathrm{E}\left[X^2\right]+\mathrm{E}[X]^2-2\mathrm{E}[X]\mathrm{E}[X] \\ &= \mathrm{E}\left[X^2\right]-\mathrm{E}[X]^2\end{aligned}$$

Άρα υπολογίζοντας:

$$E\left[Z^2\right] = 1^2\cdot\frac{1}{6}+2^2\cdot\frac{1}{6}+3^2\cdot\frac{1}{6}+4^2\cdot\frac{1}{6}+5^2\cdot\frac{1}{6}+6^2\cdot\frac{1}{6}=\frac{91}{6}$$

Οπότε η μεταβλητότητα ισούται με:

$$Var[Z] = \frac{91}{6}-3.5^2 = \frac{105}{36} \text{ ή } 2.9166667$$

Γ) Το skewness ορίζεται ως:

$$skewness = \frac{c_3}{c_2^{3/2}} = E\left[\left(\frac{x-\mu}{\sigma}\right)^3\right]$$

Και μας δείχνει την ασυμμετρία της pdf γύρω από τη μέση τιμή.

Άρα στη περίπτωση μας θα είναι :

$$E\left[\left(\frac{Z-\mu}{\sigma}\right)^3\right] = \left(\frac{1-3.5}{\sqrt{2.9166667}}\right)^3\cdot\frac{1}{6}+\left(\frac{2-3.5}{\sqrt{2.9166667}}\right)^3\cdot\frac{1}{6}+\left(\frac{3-3.5}{\sqrt{2.9166667}}\right)^3\cdot\frac{1}{6}+\left(\frac{4-3.5}{\sqrt{2.9166667}}\right)^3\cdot\frac{1}{6}$$
$$+\left(\frac{5-3.5}{\sqrt{2.9166667}}\right)^3\cdot\frac{1}{6}+\left(\frac{6-3.5}{\sqrt{2.9166667}}\right)^3\cdot\frac{1}{6}=$$
$$=\left(\left(\frac{-2.5}{\sqrt{2.9166667}}\right)^3+\left(\frac{-1.5}{\sqrt{2.9166667}}\right)^3+\left(\frac{-0.5}{\sqrt{2.9166667}}\right)^3+\left(\frac{0.5}{\sqrt{2.9166667}}\right)^3+\left(\frac{1.5}{\sqrt{2.9166667}}\right)^3\right.$$
$$\left.+\left(\frac{2.5}{\sqrt{2.9166667}}\right)^3\right)\cdot\frac{1}{6}=0$$

$$kurtosis = \frac{c_4}{c_2^2} = E\left[\left(\frac{x-\mu}{\sigma}\right)^4\right]$$

Και μας δείχνει την απόσταση από τη Γκαουσιανή κατανομή.

Άρα έχουμε:

$$E\left[\left(\frac{Z-\mu}{\sigma}\right)^4\right] = \left(\frac{1-3.5}{\sqrt{2.9166667}}\right)^4 \cdot \frac{1}{6} + \left(\frac{2-3.5}{\sqrt{2.9166667}}\right)^4 \cdot \frac{1}{6} + \left(\frac{3-3.5}{\sqrt{2.9166667}}\right)^4 \cdot \frac{1}{6} + \left(\frac{4-3.5}{\sqrt{2.9166667}}\right)^4 \cdot \frac{1}{6} + \left(\frac{5-3.5}{\sqrt{2.9166667}}\right)^4 \cdot \frac{1}{6} + \left(\frac{6-3.5}{\sqrt{2.9166667}}\right)^4 \cdot \frac{1}{6}$$

$$= \left(\left(\frac{-2.5}{\sqrt{2.9166667}}\right)^4 + \left(\frac{-1.5}{\sqrt{2.9166667}}\right)^4 + \left(\frac{-0.5}{\sqrt{2.9166667}}\right)^4 + \left(\frac{0.5}{\sqrt{2.9166667}}\right)^4 + \left(\frac{1.5}{\sqrt{2.9166667}}\right)^4 + \left(\frac{2.5}{\sqrt{2.9166667}}\right)^4\right) \cdot \frac{1}{6} =$$

$$= \left(\frac{39.0625}{2.9166667^2} + \frac{5.0625}{2.9166667^2} + \frac{0.0625}{2.9166667^2} + \frac{0.0625}{2.9166667^2} + \frac{5.0625}{2.9166667^2} + \frac{39.0625}{2.9166667^2}\right) \cdot \frac{1}{6} =$$

$$\left(\frac{88.375}{8.50694464}\right) \cdot \frac{1}{6} = 1.731428531$$

<u>Παρατήρηση:</u>

Μπορούμε να επαληθεύσουμε αυτά τα νούμερα αν λάβουμε υπόψιν τους γενικούς τύπους που προκύπτουν όταν έχουμε <u>ομοιόμορφη διακριτή κατανομή</u>, κάτι που ισχύει για την περίπτωση του ζαριού όπως είδαμε παραπάνω. Σε αυτή τη περίπτωση έχουμε ακραίες τιμές για αυτή την κατανομή a=1 και b=6 και n=b-a+1=6

Άρα

$$f(z_1) = \frac{1}{n} = \frac{1}{6}$$

$$E[z] = \frac{\alpha + b}{2} = \frac{1+6}{2} = 3.5$$

$$Var[z] = \frac{(b-a+1)^2 - 1}{12} = \frac{35}{12} = 2.91667$$

$$skewness = 0$$

$$Kurtosis = -\frac{6 \cdot \left(n^2+1\right)}{5 \cdot \left(n^2-1\right)} + 3 = -\frac{6 \cdot 37}{5 \cdot 35} + 3 = -1.26857 + 3 = 1.731428$$

Παρατηρούμε ότι προέκυψαν τα ίδια νούμερα και άρα επαληθεύονται τα αποτελέσματα των πράξεων παραπάνω.

2ο ΕΡΩΤΗΜΑ

A) Αρχικά έφτιαξα μια συνάρτηση :

```
def oneDieRolls(numOfThrows):
```

η οποία δέχεται ως όρισμα τον αριθμό των των ρίψεων που θέλουμε και επιστρέφει έναν πίνακα (numpy array) μεγέθους 1 x numOfThrows με ακεραίους από 1 έως 6

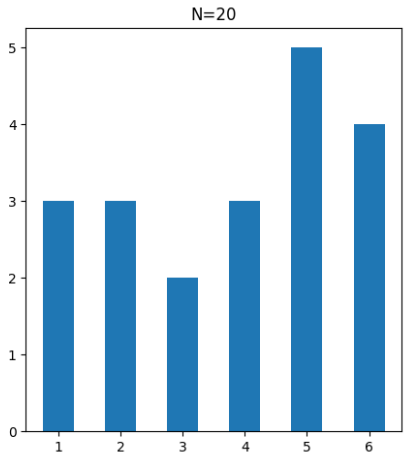
Αναλυτικά:

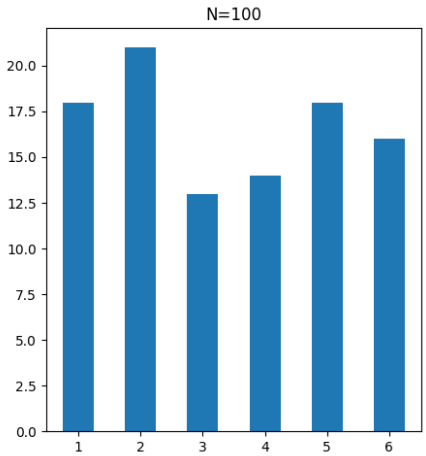
```
def oneDieRolls(numOfThrows):
    sumOfThrows= np.random.randint(low=1,high=7, size=numOfThrows)
    return sumOfThrows
```

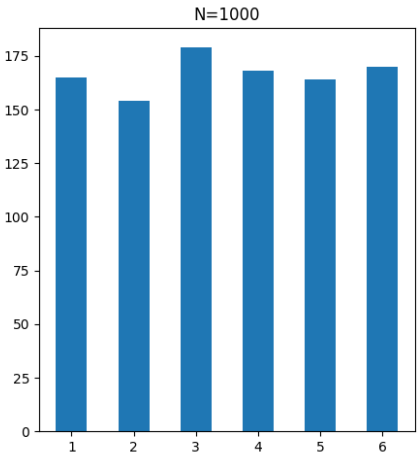
B) Με την παραπάνω συνάρτηση φτιάχνουμε τους πίνακες για N=20,100,1000 και στη συνέχεια κάνουμε το ιστόγραμμα για κάθε πίνακα με την χρήση της βιβλιοθήκης matplotlib όπως φαίνεται στο παρακάτω κώδικα:

```
d = np.diff(np.unique(listOfThrows[0])).min()
left_of_first_bin =listOfThrows[0].min() - float(d)/2
right_of_last_bin = listOfThrows[0].max() + float(d)/2
fig, axes = plt.subplots(nrows=1, ncols=3)
ax0, ax1, ax2 = axes.flatten()
ax0.hist(listOfThrows[0], np.arange(left_of_first_bin, right_of_last_bin + d,
d),rwidth=0.5)
ax0.set_title('N=20')
ax1.hist(listOfThrows[1], np.arange(left_of_first_bin, right_of_last_bin + d,
d),rwidth=0.5)
ax1.set_title('N=100')
ax2.hist(listOfThrows[2], np.arange(left_of_first_bin, right_of_last_bin + d,
d),rwidth=0.5)
ax2.set_title('N=1000')
fig.tight_layout()
plt.show()
```

και έχουμε τα εξής αποτελέσματα:







λογικό και ακολουθεί τον Νόμο των μεγάλων αριθμών. Δηλαδή όσο αυξάνουμε τον αριθμό τον πειραμάτων τόσο η πραγματική πιθανότητα να συμβεί ένα ενδεχόμενο προσεγγίζει την θεωρητική, καθώς και ο δειγματικός μέσος μιας ακολουθίας ανεξάρτητων τυχαίων μεταβλητών που ακολουθούν μία κοινή κατανομή συγκλίνει προς τον θεωρητικό μέσο (ή μέση τιμή) της κατανομής.

Γ) Τώρα υπολογίζουμε για N=10,20,50,100,500,1000 τη μέση τιμή, τη μεταβλητότητα, το skewness και την kurtosis για κάθε N χρησιμοποιώντας τις αντίστοιχες συναρτήσεις της βιβλιοθήκης NumPy και SciPy.

(ΠΑΡΑΤΗΡΗΣΗ: Εδώ θέλει προσοχή αναφορικά με την συνάρτηση kurtosis της SciPy καθώς έχει default παράμετρο να υπολογίζει την Excess Kurtosis δηλαδή την kurtosis που υπολογίζουμε εμείς -3 (Αυτό γίνεται για να έχει η κανονική κατανομή kurtosis 0 και όχι 3) οπότε για να έχουμε τον ίδιο τύπο με αυτόν που χρησιμοποιούμε στη θεωρία πρέπει να βάλουμε την παράμετρο **fisher=True**)

Με τον παρακάτω κώδικα υπολογίζουμε και τυπώνουμε τα αποτελέσματα:

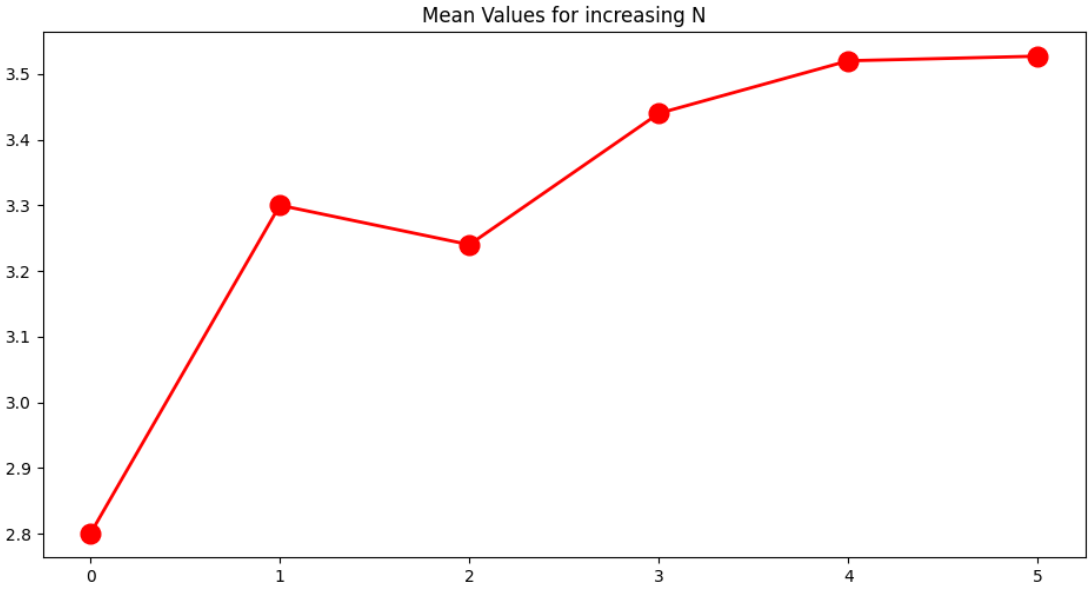
```
nValues=[10,20,50,100,500,1000]
newListOfThrows=[]
np.random.seed(9990)
for n in nValues:
   newListOfThrows.append(oneDieRolls(n))
meanValue=[]
varianceValue=[]
skewnessValue=[]
kurtosisValue=[]
#Mean value calculation for diff N values (τα εχω σε μορφή πίνακα μήπως χρειαστεί κάποιο plot)
for i in range(len(newListOfThrows)):
   meanValue.append(np.mean(newListOfThrows[i]))
   varianceValue.append(np.var(newListOfThrows[i]))
   skewnessValue.append(skew(newListOfThrows[i]))
   kurtosisValue.append(kurtosis(newListOfThrows[i],fisher=False))
   print(20 * '==')
   print("FOR",nValues[i],"DIE ROLLS")
   print("Mean value: ",format(meanValue[i],'.4f')," Difference: ",format(abs(3.5-
meanValue[i]),'.4f'))
   print("Variance: ",format(varianceValue[i],'.4f')," Difference: ",format(abs(2.9166-
varianceValue[i]),'.4f'))
   print("Skewness is",format(skewnessValue[i],'.4f'),"Difference:
",format(abs(skewnessValue[i]),'.4f'))
   print("Kurtosis is",format(kurtosisValue[i],'.4f'),"Difference:
",format(abs(1.731428-kurtosisValue[i]),'.4f'))
```

και έχω τα εξής αποτελέσματα:

```
D:\PatternRec1\venv\Scripts\python.exe D:/PatternRec1/Ex1.py
====================================
FOR 10 DIE ROLLS
Mean value:  2.8000  Difference:  0.7000
Variance:  2.7600  Difference:  0.1566
Skewness is 0.5810 Difference:  0.5810
Kurtosis is 2.1355 Difference:  0.4040
====================================
FOR 20 DIE ROLLS
Mean value:  3.3000  Difference:  0.2000
Variance:  2.3100  Difference:  0.6066
Skewness is 0.5964 Difference:  0.5964
Kurtosis is 2.1049 Difference:  0.3734
====================================
FOR 50 DIE ROLLS
Mean value:  3.2400  Difference:  0.2600
Variance:  3.2224  Difference:  0.3058
Skewness is 0.1981 Difference:  0.1981
Kurtosis is 1.5585 Difference:  0.1730
====================================
====================================
FOR 100 DIE ROLLS
Mean value:  3.4400  Difference:  0.0600
Variance:  3.1064  Difference:  0.1898
Skewness is 0.0720 Difference:  0.0720
Kurtosis is 1.6705 Difference:  0.0609
====================================
FOR 500 DIE ROLLS
Mean value:  3.5200  Difference:  0.0200
Variance:  2.8856  Difference:  0.0310
Skewness is -0.0527 Difference:  0.0527
Kurtosis is 1.7575 Difference:  0.0261
====================================
FOR 1000 DIE ROLLS
Mean value:  3.5270  Difference:  0.0270
Variance:  2.8113  Difference:  0.1053
Skewness is -0.0272 Difference:  0.0272
Kurtosis is 1.7660 Difference:  0.0346
```

Δ) Για 500 ρίψεις και πάνω στην συγκεκριμένη περίπτωση αρχίζουμε να προσεγγίζουμε τις θεωρητικές τιμές πολύ καλά (βέβαια εδώ και στις 100 ρίψεις με εξαίρεση το variance έχουμε πολύ καλή προσέγγιση).

Ε) Μια στοχαστική διεργασία είναι Stationary, όταν οι στατιστικές της ιδιότητες είναι ανεξάρτητες σε αλλαγές στο πεδίο ορισμού της. Wide Sense Stationarity (WSS) χαρακτηρίζεται συγκεκριμένα, εάν η μέση τιμή της είναι αυτή που μένει σταθερή. Εδώ αν κάνουμε plot τα means για κάθε διαφορετικό N από το μικρότερο στο μεγαλύτερο:



Παρατηρούμε ότι για N=500 και μετά έχουμε πολύ μικρή διαφορά στο mean value οπότε μπορούμε να το θεωρήσουμε σχεδόν σταθερό οπότε για N=500 ξεκινάμε να έχουμε wide-sense stationarity.

3ο ΕΡΩΤΗΜΑ

α) Θεωρητικά η κοινή συνάρτηση πυκνότητας για τις ανεξάρτητες μεταξύ τους μεταβλητές z1 και z2 είναι:

$$f(z_1, z_2) = \frac{1}{6} \cdot \frac{1}{6} = \frac{1}{36}$$

Μπορούμε να το δούμε και έτσι:

$X \backslash Y$	1	2	3	4	5	6
1	1/36	1/36	1/36	1/36	1/36	1/36
2	1/36	1/36	1/36	1/36	1/36	1/36
3	1/36	1/36	1/36	1/36	1/36	1/36
4	1/36	1/36	1/36	1/36	1/36	1/36
5	1/36	1/36	1/36	1/36	1/36	1/36
6	1/36	1/36	1/36	1/36	1/36	1/36

Πειραματικά θα δημιουργήσουμε 1000 ρίψεις για z1 και 1000 για z2

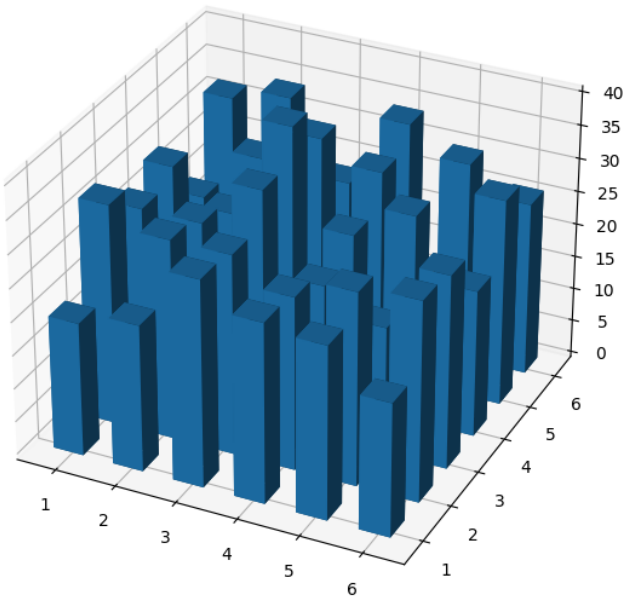
```
np.random.seed(2)
z1=np.random.randint(low=1,high=7, size=1000)
z2=np.random.randint(low=1,high=7, size=1000)
```

και θα κάνουμε ένα ιστόγραμμα 2 διαστάσεων για να αναπαραστήσουμε έτσι όλα τα ζεύγη ξεχωριστά (πχ {1,2},{1,1},{2,1} κτλ.)

(αναλυτικά ο κώδικας για το 2d histogram μέσω του matplotlib και της NumPy βρίσκεται ex1-2Dice.py)

Βλέπουμε λοιπόν:

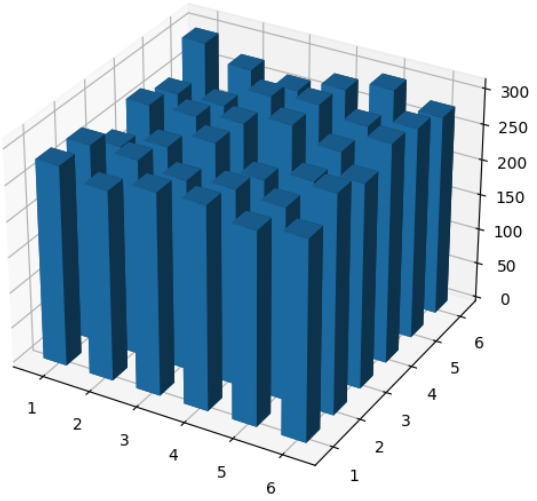
Για N=1000



Εδώ βλέπουμε ότι υπάρχει μια μικρή προσέγγιση προς την θεωρητική τιμή

(δοκίμασα και με N=10000 για να δούμε καλύτερη προσέγγιση με το αποτέλεσμα:

Για N=10000



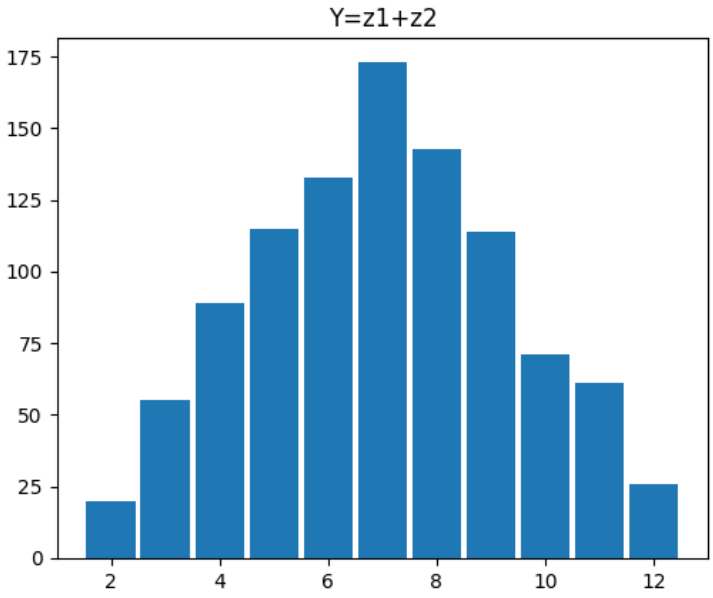
Εδώ βλέπουμε ακόμα μεγαλύτερη ομοιομορφία)

Β) Τώρα αθροίζουμε τα z1,z2 και κάνουμε το ιστόγραμμα για τιμές αυτές (το άθροισμα μπορεί να είναι από 2 έως 12)

Με τον παρακάτω κώδικα:

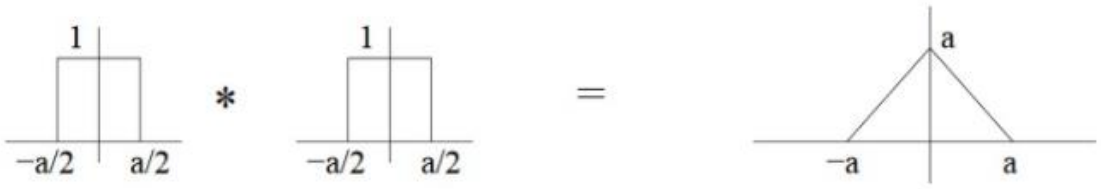
```
y=z1+z2
plt.hist(y, bins=np.arange(2, 14), align="left", rwidth=0.9)
plt.title("Y=z1+z2")
plt.show()
```

Και έχουμε τα εξής αποτελέσματα:



Το αποτέλεσμα αυτό είναι απολύτως λογικό με βάση την ιδιότητα των pdf ότι αν y =z1+z2, τότε f(y)= f(z1)*f(z2), (όπου * ο τελεστής της συνέλιξης)

Πιο συγκεκριμένα αν παρατηρήσουμε τις δύο αυτές μεταβλητές που συνελίσσονται, θυμόμαστε ότι οι ξεχωριστές τους pdf ακολουθούν την ομοιόμορφη διακριτή κατανομή. Άρα εμφανίζουν την μορφή ενός ορθογώνιου παλμού. Όπως είχαμε δει και στο μάθημα της ψηφιακής επεξεργασίας σήματος στην περίπτωση που έχουμε 2 <u>όμοιους</u> ορθογώνιους παλμούς η συνέλιξη τους θα έχει την μορφή ενός τριγωνικού παλμού:



ΒΕΛΤΙΣΤΟΠΟΙΗΣΗ ΧΩΡΙΣ ΠΕΡΙΟΡΙΣΜΟΥΣ

<u>4ο ΕΡΩΤΗΜΑ</u>

Α) Με βάση την ιδιότητα της παραγώγου βρίσκουμε:

$$f(x) = (x-5)^4 + 3x$$

$$f'(x) = 4(x-5)^3 + 3$$

$$f'(x) = 0 \Longrightarrow$$
$$\Longrightarrow 4(x-5)^3 + 3 = 0 \Longrightarrow$$
$$\Longrightarrow (x-5)^3 = -\frac{3}{4} \Longrightarrow \quad (\gamma\iota\alpha\ x\ \pi\rho\alpha\gamma\mu\alpha\tau\iota\kappa o\acute{\upsilon}\varsigma)$$
$$\Longrightarrow x-5 = -\sqrt[3]{\frac{3}{4}} \Longrightarrow x = -\sqrt[3]{\frac{3}{4}} + 5 = 4.0914$$

$$f''(x) = 12(x-5)^2 \geqslant 0\ (\eta\ \iota\sigma\acute{o}\tau\eta\tau\alpha\ \iota\sigma\chi\acute{\upsilon}\epsilon\iota\ \gamma\iota\alpha\ x = 5)$$
$$\acute{\alpha}\rho\alpha\ f''(4.0914) > 0\ \acute{\alpha}\rho\alpha\ x = 4.0914\ o\lambda\iota\kappa\acute{o}\ \epsilon\lambda\acute{\alpha}\chi\iota\sigma\tau o\ \tau\eta\varsigma\ \sigma\upsilon\nu\acute{\alpha}\rho\tau\eta\sigma\eta\varsigma$$

Β) Δημιούργησα την συνάρτηση:

```
def gradientDescent(df,starting_x,learningRate,maxIters,minStep):
```

όπου τα ορίσματα είναι

df: Η παράγωγος της συνάρτησης που θέλω να ελαχιστοποιήσω

starting_x: Αρχικό σημείο x.

learningRate: Βήμα n.

minStep: Η ελάχιστη διαφοροποίηση που πρέπει να κάνει το x από ένα βήμα στο επόμενο. Αν η αλλαγή του x είναι μικρότερη από το minStep, τότε σταματάμε.

maxIters: Οι μέγιστες επιτρεπόμενες επαναλήψεις του αλγορίθμου.

Υλοποιώ το σώμα της συνάρτησης με βάση την επαναληπτική μέθοδο που είδαμε:

```
iters=0
cur_x=starting_x
step=100 #dummy value for first iteration
while  iters < maxIters and abs(step)>minStep:
   prev_x = cur_x  # Store current x value in prev_x
   cur_x = prev_x - learningRate * df(prev_x)  # Grad descent
   iters = iters + 1  # iteration count
   step= prev_x-cur_x

print("The local minimum is at", cur_x,'after',iters,'iterations')
return cur_x, iters
```

Διαλέγουμε αρχικό x=10. Και καλούμε τη συνάρτηση:

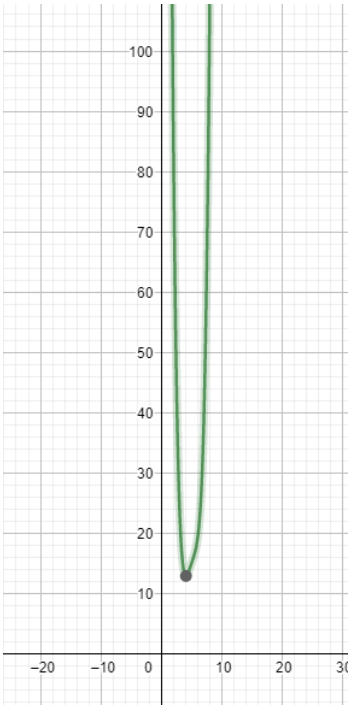
```
df=lambda x: 4*pow((x-5),3)+3
gradientDescent(df,10,0.01,1000,1e-9)
```

Κάνοντας διάφορες δοκιμές έχω αυτό το αποτέλεσμα

```
D:\PatternRec1\venv\Scripts\python.exe D:/PatternRec1/ex1-GradientDescend.py
The local minimum is at 4.0914397125409385 after 192 iterations
```

Παρατήρηση

Σε αυτήν την μέθοδο είναι σημαντικό να δούμε μερικές περιπτώσεις. Δηλαδή με βάση το πως είναι γραφικά η συνάρτηση μας:



Στο σημείο x=10 έχουμε πολύ μεγάλη κλίση. Αν εκεί διαλέξουμε ένα μεγάλο n (learningRate) τότε θα έχουμε την τιμή του x να «εκτοξεύεται» καθώς το βήμα με την μεγάλη κλίση «οδηγεί» πολύ αριστερά το x. Έπειτα, επειδή έχει πάλι μεγάλη κλίση θα πάει πολύ δεξιά στο επόμενο βήμα και αυτό επαναλαμβάνεται μέχρι που έχουμε κάποιο overflow:

```
D:\PatternRec1\venv\Scripts\python.exe D:/PatternRec1/ex1-GradientDescend.py
Traceback (most recent call last):
  File "D:/PatternRec1/ex1-GradientDescend.py", line 16, in <module>
    gradientDescent(df,10,0.02,1000,1e-9)
  File "D:/PatternRec1/ex1-GradientDescend.py", line 7, in gradientDescent
    cur_x = prev_x - learningRate * df(prev_x)  # Grad descent
  File "D:/PatternRec1/ex1-GradientDescend.py", line 15, in <lambda>
    df=lambda x: 4*pow((x-5),3)+3
OverflowError: (34, 'Result too large')

Process finished with exit code 1
```

Β) Τώρα θα δοκιμάσουμε την μέθοδο Newton.

Δημιούργησα την εξής συνάρτηση:

```
def newtonMethod(firstDeriv,secondDeriv,starting_x,maxIters,minStep):
```

όπου

firstDeriv: Η πρώτη παράγωγος της συνάρτησης μας.

secondDeriv: Η δεύτερη παράγωγος της συνάρτησης μας.

starting_x: Το αρχικό x για την μέθοδο.

minStep: Η ελάχιστη διαφοροποίηση που πρέπει να κάνει το x από ένα βήμα στο επόμενο. Αν η αλλαγή του x είναι μικρότερη από το minStep, τότε σταματάμε.

maxIters: Οι μέγιστες επιτρεπόμενες επαναλήψεις.

Και υλοποιούμε την μέθοδο:

```
iters=0
cur_x=starting_x
step=100
while iters<maxIters and abs(step)>minStep:
   prev_x=cur_x
   h = firstDeriv(cur_x) /secondDeriv(cur_x)
   cur_x= cur_x - h
   iters = iters + 1  # iteration count
   step= prev_x-cur_x

print("The local minimum occurs at", cur_x,'after',iters,"iterations")
return cur_x,iters
```

Καλούμε για ίδιο αρχικό x=10:

```
df=lambda x: 4*pow((x-5),3)+3
ddf=lambda x:12*pow((x-5),2)

newtonMethod(df,ddf,10,100,1e-9)
```

Βλέπουμε ότι φτάνουμε στο ελάχιστο σε πολύ λιγότερες επαναλήψεις:

```
D:\PatternRec1\venv\Scripts\python.exe D:/PatternRec1/ex1-Newton.py
The local minimum occurs at 4.09143970358393 after 17 iterations
```

Το αποτέλεσμα αυτό είναι λογικό καθώς η μέθοδος newton χρησιμοποιεί και την καμπυλότητα της συνάρτησης ως παραπάνω πληροφορία και έτσι συγκλίνει γρηγορότερα. Επίσης δεν αντιμετώπισε το πρόβλημα που περιγράψαμε παραπάνω στο learningRate όταν έχω μεγάλη κλίση.