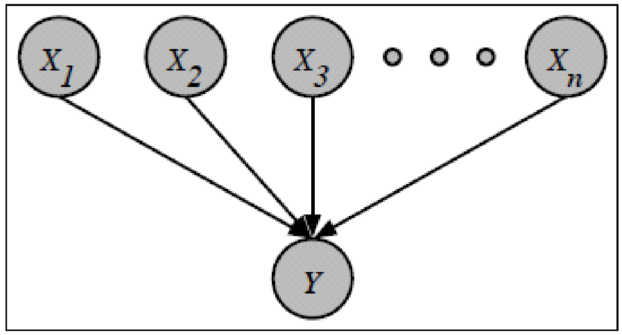
6.2 EM algorithm for noisy-OR

Consider the belief network on the right, with binary random variables $X \in \{0,1\}^n$ and $Y \in \{0,1\}$ and a noisy-OR conditional probability table (CPT). The noisy-OR CPT is given by:

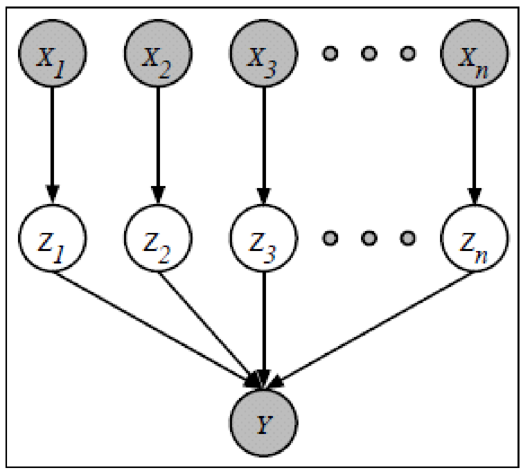


$$P(Y=1|X) \;=\; 1-\prod_{i=1}^{n}(1-p_i)^{X_i},$$

which is expressed in terms of the noisy-OR parameters $p_i \in [0,1]$.

In this problem, you will derive and implement an EM algorithm for estimating the noisy-OR parameters p_i. It may seem that the EM algorithm is not suited to this problem, in which all the nodes are observed, and the CPT has a parameterized form. In fact, the EM algorithm can be applied, but first we must express the model in a different but equivalent form.

Consider the belief network shown to the right. In this network, a binary random variable $Z_i \in \{0,1\}$ intercedes between each pair of nodes X_i and Y. Suppose that:



$$\begin{aligned} P(Z_i=1|X_i=0) &= 0,\\ P(Z_i=1|X_i=1) &= p_i. \end{aligned}$$

Also, let the node Y be *determined* by the logical-OR of Z_i. In other words:

$$P(Y=1|Z) \;=\; \begin{cases} 1 \text{ if } Z_i=1 \text{ for any } i,\\ 0 \text{ if } Z_i=0 \text{ for all } i. \end{cases}$$

(a) Show that this "extended" belief network defines the same conditional distribution $P(Y|X)$ as the original one. In particular, starting from

$$P(Y=1|X) \;=\; \sum_{Z\in\{0,1\}^n} P(Y=1,Z|X),$$

show that the right hand side of this equation reduces to the noisy-OR CPT with parameters p_i. To perform this marginalization, you will need to exploit various conditional independence relations.

Pf: $P(Y=1|X) \underset{\text{marginalization}}{=} \sum_{Z\in\{0,1\}^n} P(Y=1,Z|X) \overset{\text{product rule}}{=} \sum_{Z\in\{0,1\}^n} P(Y=1|Z,X)\,P(Z|X)$

|| conditional indep from the BN

$\sum_{Z\in\{0,1\}^n} P(Y=1|Z)\,P(Z|X)$

Since we know:

Since we have:

$$P(Y=1|Z) = \begin{cases} 1 \text{ if } Z_i = 1 \text{ for any } i, \\ 0 \text{ if } Z_i = 0 \text{ for all } i. \end{cases}$$

So we can write: $P(Y=1|Z) = 1 - \prod_{i=1}^{n} (1-Z_i)$

So $P(Y=1|X) = \sum_{Z\in\{0,1\}^n} P(Y=1|Z)P(Z|X) = \sum_{Z\in\{0,1\}^n} \left(1-\prod_{i=1}^{n}(1-Z_i)\right) P(Z|X)$

$$= \sum_{Z\in\{0,1\}^n} P(Z|X) - \sum_{Z\in\{0,1\}^n} P(Z|X)\left(\prod_{i=1}^{n}(1-Z_i)\right)$$

$$\stackrel{\text{marginalization}}{=} 1 - \sum_{Z\in\{0,1\}^n} P(Z|X)\left(\prod_{i=1}^{n}(1-Z_i)\right)$$

Since Y d-separates $\{X_i, Z_i\}$ from $\{X_j, Z_j\}$ (for $i \neq j$)

$\Downarrow$

$P(Z|X) = \prod_{i=1}^{n} P(Z_i|X_i)$

So we have: $P(Z|X)\left(\prod_{i=1}^{n}(1-Z_i)\right) = \prod_{i,j=1}^{n} P(Z_i|X_i)(1-Z_j)$

Now, we have:

$$\left.\begin{aligned} P(Z_i=1|X_i=0) &= 0, \\ P(Z_i=1|X_i=1) &= p_i. \end{aligned}\right\} \Rightarrow P(Z_i|X_i) = \begin{cases} 1-(1-p_i)^{X_i} & \text{if } Z_i = 1 \\ (1-p_i)^{X_i} & \text{if } Z_i = 0 \end{cases}$$

So, $\prod_{i,j=1}^{n} P(Z_i|X_i)(1-Z_j) = \begin{cases} \prod_{i=1}^{n}(1-p_i)^{X_i} & \text{if all } Z_j\text{s} = 0 \\ 0 & \text{otherwise} \end{cases}$

So

$$P(Y=1|X) = 1 - \sum_{Z\in\{0,1\}^n} P(Z|X)\left(\prod_{i=1}^{n}(1-Z_i)\right) = 1 - P(Z|X)\prod_{i=1}^{n}(1-Z_i)\Big|_{Z=(Z_1,\dots,Z_n)=(0,\dots,0)}$$

$$= 1 - \prod (1-p_i)^{X_i}$$

i=1

(Proved)

(b) Consider estimating the noisy-OR parameters p_i to maximize the (conditional) likelihood of the observed data. The (normalized) log-likelihood in this case is given by:

$$\mathcal{L} = \frac{1}{T}\sum_{t=1}^{T} \log P(Y=y^{(t)}|X=\vec{x}^{(t)}),$$

where $(\vec{x}^{(t)}, y^{(t)})$ is the tth joint observation of X and Y, and where for convenience we have divided the overall log-likelihood by the number of examples T. From your result in part (a), it follows that we can estimate the parameters p_i in either the original network or the extended one (since in both networks they would be maximizing the same equation for the log-likelihood).

Notice that in the extended network, we can view X and Y as observed nodes and Z as hidden nodes. Thus in this network, we can use the EM algorithm to estimate each parameter p_i, which simply defines one row of the "look-up" CPT for the node Z_i.

Compute the posterior probability that appears in the E-step of this EM algorithm. In particular, for joint observations $x \in \{0,1\}^n$ and $y \in \{0,1\}$, use Bayes rule to show that:

$$P(Z_i=1, X_i=1|X=x, Y=y) = \frac{y x_i p_i}{1-\prod_j (1-p_j)^{x_j}}$$

$x_i^{(t)}$ = x_i's value in tth trial

Ans: The posterior probabilities:

$$\left.\begin{aligned} &P(X_i = x_i \mid Y=y^{(t)}, X=x^{(t)}) = I(x_i, x_i^{(t)}) \\ &\& \; P(Y=y \mid Y=y^{(t)}, X=x^{(t)}) = I(y, y^{(t)}) \end{aligned}\right\} \text{(Ans)}$$

Also, we have $P(Z_i=1, X_i=1 \mid X=x, Y=y) = I(1, x_i)\, P(Z_i=1 \mid X=x, Y=y)$

$\| \text{Bayes' rule}$

$$\frac{P(Z_i=1|X=x)\; I(1,x_i)\; P(Y=y|X=x, Z_i=1)}{P(Y=y|X=x)}$$

// conditional indep from BN above

$$\frac{P(Z_i=1|X_i=x_i)\, I(1,x_i)\, P(Y=y|Z_i=1, X=x)}{P(Y=y|X=x)}$$

Now $P(Y=1|Z)=0$ if all Z_is $=0$ & $P(Y=1|Z)=1$ otherwise.

So $P(Y=y|X=x, Z_i=1) = \begin{cases} 0 & \text{if } y=0 \\ 1 & \text{if } y=1 \end{cases} = I(1,y) = y$ (as $y \in \{0,1\}$)

So $P(Z_i=1, X_i=1|Y=y, X=x) = y\, I(1,x_i) \dfrac{P(Z_i=1|X_i=x_i)}{P(Y=y|X=x)}$

where $P(Y=y|X=x) = 1-\prod_{i=1}^{n}(1-p_i)^{x_i}$ (given)

So

$$P(Z_i=1, X_i=1|Y=y, X=x) = y\, I(1,x_i) \frac{P(Z_i=1|X_i=x_i)}{1-\prod_{j=1}^{n}(1-p_j)^{x_j}}$$

as $x_i \in \{0,1\}$, $I(1,x_i) = x_i$

$$= y x_i \frac{P(Z_i=1|X_i=x_i)}{1-\prod_{j=1}^{n}(1-p_j)^{x_j}}$$

Now, $P(Z_i=1|X_i=x_i) = \begin{cases} 0 & \text{if } x_i=0 \\ p_i & \text{if } x_i=1 \end{cases} = \underbrace{I(x_i,1)}_{x_i} p_i = x_i p_i$ (as $x_i \in \{0,1\}$)

So $P(Z_i=1, X_i=1|Y=y, X=x) = \dfrac{y x_i \; x_i p_i}{1-\prod_{j=1}^{n}(1-p_j)^{x_j}} \underset{\substack{\text{since} \\ x_i\in\{0,1\} \\ x_i^2=x_i}}{=} \dfrac{y x_i p_i}{1-\prod_{j=1}^{n}(1-p_j)^{x_j}}$ (proved).

(c) For the data set $\{\vec{x}^{(t)}, y^{(t)}\}_{t=1}^{T}$, show that the EM update for the parameters p_i is given by:

$$p_i \leftarrow \frac{1}{T_i}\sum_t P\left(Z_i=1, X_i=1|X=x^{(t)}, Y=y^{(t)}\right),$$

where T_i is the number of examples in which $X_i=1$. (You should derive this update as a special case of the general form presented in lecture.)

Pf: we know from lecture, in EM update for nodes with parents, we have: $P(x_i = x \mid Pa_i = \pi) \leftarrow \dfrac{\sum_t P(X_i = x, Pa_i = \pi \mid V_t = v_t)}{\sum_t P(Pa_i = \pi \mid V_t = v_t)}$

We have, $p_i = P(z_i = 1 \mid X_i = 1)$, so the EM update for p_i is:

$$p_i \leftarrow \frac{\sum_t P(z_i = 1, X_i = 1 \mid X = x^{(t)}, Y = y^{(t)})}{\sum_t P(X_i = 1 \mid X = x^{(t)}, Y = y^{(t)})}$$

$$\parallel$$

$$\frac{1}{\sum_t I(1, x_i^{(t)})} \sum_t P(z_i = 1, X_i = 1 \mid X = x^{(t)}, Y = y^{(t)})$$

here $\sum_t I(1, x_i^{(t)})$ = no. of times X_i is nonzero $(=1)$ in the samples $= T_i$ (by defn)

So we have $\boxed{p_i \leftarrow \frac{1}{T_i} \sum_t P(z_i = 1, X_i = 1 \mid X = x^{(t)}, Y = y^{(t)})}$ ← as our EM update step (proved)

(d) Download the data files on the course web site, and use the EM algorithm to estimate the parameters p_i. The data set[1] has $T = 267$ examples over $n = 23$ inputs. To check your solution, initialize all $p_i = 0.05$ and perform 256 iterations of the EM algorithm. At each iteration, compute the log-likelihood shown in part (b). (If you have implemented the EM algorithm correctly, this log-likelihood will always increase from one iteration to the next.) Also compute the number of mistakes $M \leq T$ made by the model at each iteration; a mistake occurs either when $y_t = 0$ and $P(y_t = 1|\vec{x}_t) \geq 0.5$ (indicating a false positive) or when $y_t = 1$ and $P(y_t = 1|\vec{x}_t) \leq 0.5$ (indicating a false negative). The number of mistakes should generally decrease as the model is trained, though it is not guaranteed to do so at each iteration. Complete the following table:

iteration	number of mistakes M	log-likelihood $\mathcal{L}$
0	175	-0.95809
1	56	
2		-0.40822
4		
8		
16		
32		
64	37	
128		
256		-0.31016

You may use the already completed entries of this table to check your work.

Ans: My completes table looks like:

iteration	number of mistakes M	log-likelihood $\mathcal{L}$
0	175	-0.9580854082157914
1	56	-0.49591639407753635
2	43	-0.40822081705839114
4	42	-0.3646149825001877.
8	44	-0.34750061620878253
16	40	-0.33461704895854844
32	37	-0.32258140316749784
64	37	-0.3148266983628559
128	36	-0.3111558472151897
256	36	-0.310161353474076

(e) Turn in your source code. As always, you may program in the language of your choice.

[1] For those interested, more information about this data set is available at *http://archive.ics.uci.edu/ml/datasets/SPECT+Heart*. However, be sure to use the data files provided on Canvas, as they have been specially assembled for this assignment.

Ans: Please find the source code in the following:

Homework 6 Problem 2

November 5, 2022

1 Source Code and outputs for Problem 6.2(d)_Hw6_CSE 250A_Fall 2022

[5]:
```
import numpy as np
import matplotlib.pyplot as plt
import copy

def intlistconvert(strlist):    #function to convert a character list to an␣
 ↪integer list, if applicable
    intlist=[int(stringel) for stringel in strlist]
    return intlist

##########reading X.txt and Y.txt␣
 ↪files############################################################
xdatalist=[]
with open('x.txt') as f:
    for line in f:
        xdatalist.append(list(line.strip().replace(" ", "")))

T=len(xdatalist)    # no. of trials

xdatalist=[intlistconvert(el) for el in xdatalist]
xdata=np.array(xdatalist)    # first index here will store trial number and␣
 ↪second index will store index 'i' of 'x_i'
n=len(xdata[0,:])   # no. of variables X_i s

ydatalist=[]
with open('y.txt') as f:
    for line in f:
        ydatalist.append(line.strip())

ydatalist=intlistconvert(ydatalist)
ydata=np.array(ydatalist)


# forming the T_i s
Tis=np.zeros(n, dtype=int)
```

```python
for i in range(n):
    Tis[i]=np.count_nonzero(xdata[:,i])  # counts no. of nonzero (or 1) entries for each X_i in the training data


############ start iterations for EM algorithm ################################################################
############################## defining relevant functions for the iterations
def prod(t,problist):
    pro=1
    for i in range(n):
        pro=pro*np.power((1-problist[i]),xdata[t,i])
    return pro


def loglikelihood(problist):
    L=0
    for t in range(T):
        pro=prod(t,problist)
        if ydata[t]==1:
            L=L+np.log(1-pro)
        else:
            L=L+np.log(pro)
    return L/T

def mistakes(problist):
    mist=0
    for t in range(T):
        pro=prod(t,problist)
        py1x=1-pro

        if (ydata[t]==0 and py1x>=0.5) or (ydata[t]==1 and py1x<=0.5):
            mist=mist+1
    return mist

# the main iteration for EM algorithm
Loglik=[]      #stores log likelihood for each iteration
M=[]           # stores number of mistakes for each iteration
for itera in range(257):
    if itera==0:
        p=np.full(n, 0.05)    # first array of probabilities p_is
    else:
        f=copy.deepcopy(p)
        for i in range(n):
            sum=0
            for t in range(T):
                sum=sum+(ydata[t]*xdata[t,i]*f[i])/(1-prod(t,f))
```

```python
            p[i]=sum/Tis[i]
    m=mistakes(p)
    logp=loglikelihood(p)

    if itera in [0,1,2,4,8,16,32,64,128,256]:
        print(f"For iteration {itera},")
        print(f"no. of mistakes is {m} and")
        print(f"(normalized)log likelihood is {logp}.\n\n")


    M=M+[m]    # creates the list of mistakes as iteration goes
    Loglik=Loglik+[logp]    # creates the log likelihood (normalized) list as␣
↪iteration goes
```

```
For iteration 0,
no. of mistakes is 175 and
(normalized)log likelihood is -0.9580854082157914.


For iteration 1,
no. of mistakes is 56 and
(normalized)log likelihood is -0.49591639407753635.


For iteration 2,
no. of mistakes is 43 and
(normalized)log likelihood is -0.40822081705839114.


For iteration 4,
no. of mistakes is 42 and
(normalized)log likelihood is -0.3646149825001877.


For iteration 8,
no. of mistakes is 44 and
(normalized)log likelihood is -0.34750061620878253.


For iteration 16,
no. of mistakes is 40 and
(normalized)log likelihood is -0.33461704895854844.


For iteration 32,
no. of mistakes is 37 and
(normalized)log likelihood is -0.32258140316749784.
```

```
For iteration 64,
no. of mistakes is 37 and
(normalized)log likelihood is -0.3148266983628559.


For iteration 128,
no. of mistakes is 36 and
(normalized)log likelihood is -0.3111558472151897.


For iteration 256,
no. of mistakes is 36 and
(normalized)log likelihood is -0.310161353474076.
```

[7]:
```
print(f"the list of mistakes (with iterations) is: {M} \n\n")
print(f"the list of normalized Log-likelihoods (with iterations) is: {Loglik}")
```

```
the list of mistakes (with iterations) is: [175, 56, 43, 40, 42, 44, 44, 44, 44,
44, 42, 42, 41, 39, 39, 40, 40, 40, 40, 39, 39, 39, 38, 38, 38, 38, 38, 38, 38,
38, 37, 37, 37, 37, 37, 37, 37, 37, 37, 37, 37, 37, 37, 37, 37, 37, 37, 37, 37,
37, 37, 37, 37, 37, 37, 37, 37, 37, 37, 37, 37, 37, 37, 37, 37, 37, 37, 37, 37,
37, 37, 37, 37, 37, 37, 37, 37, 37, 37, 37, 37, 37, 37, 37, 37, 37, 37, 37, 37,
37, 37, 37, 37, 37, 37, 37, 37, 37, 37, 37, 37, 37, 37, 37, 37, 37, 37, 37, 37,
37, 37, 37, 37, 37, 37, 36, 36, 36, 36, 36, 36, 36, 36, 36, 36, 36, 36, 36, 36,
36, 36, 36, 36, 36, 36, 36, 36, 36, 36, 36, 36, 36, 36, 36, 36, 36, 36, 36, 36,
36, 36, 36, 36, 36, 36, 36, 36, 36, 36, 36, 36, 36, 36, 36, 36, 36, 36, 36, 36,
36, 36, 36, 36, 36, 36, 36, 36, 36, 36, 36, 36, 36, 36, 36, 36, 36, 36, 36, 36,
36, 36, 36, 36, 36, 36, 36, 36, 36, 36, 36, 36, 36, 36, 36, 36, 36, 36, 36, 36,
36, 36, 36, 36, 36, 36, 36, 36, 36, 36, 36, 36, 36, 36, 36, 36, 36, 36, 36, 36,
36, 36, 36, 36, 36, 36, 36, 36, 36, 36, 36, 36, 36, 36, 36, 36, 36, 36, 36, 36,
36, 36, 36, 36, 36, 36, 36, 36]


the list of normalized Log-likelihoods (with iterations) is:
[-0.9580854082157914, -0.49591639407753635, -0.40822081705839114,
-0.3779406836061008, -0.3646149825001877, -0.35757532996649616,
-0.3531842166828877, -0.3500255657709249, -0.34750061620878253,
-0.3453400990924291, -0.3434159529651389, -0.3416634501406622,
-0.34004737198005874, -0.3385467585599132, -0.3371478191245712,
-0.33584054521650464, -0.33461704895854844, -0.33347072395698485,
-0.33239580702080124, -0.3313871394626801, -0.3304400301620283,
-0.3295501719703101, -0.32871358703290604, -0.32792658843810424,
-0.3271857515339723, -0.32648789127539796, -0.325830043534388,
-0.32520944914451916, -0.32462353991248716, -0.32406992609548135,
-0.3235463850039954, -0.3230508504922716, -0.32258140316749784,
```

-0.3221362611969939, -0.321713771627242, -0.32131240215379475,
-0.3209307332995044, -0.32056745097177336, -0.32022133937891256,
-0.31989127429209774, -0.3195762166435478, -0.3192752064539766,
-0.3189873570835853, -0.318711849801137, -0.3184479286653898,
-0.31819489571249177, -0.3179521064420621, -0.3177189655937281,
-0.3174949232049489, -0.3172794709400749, -0.31707213867983924,
-0.31687249135984846, -0.3166801260461732, -0.3164946692358l027,
-0.31631577436960484, -0.31614311954518837, -0.315976405417543,
-0.315815353275018, -0.3156597032788655, -0.31550921285474215,
-0.3153636552250264, -0.3152228180712771, -0.3150865023166463,
-0.3149545210186029, -0.3148266983628559, -0.31470286874992826,
-0.31458287596635803, -0.3144665724330886, -0.3143538185240861,
-0.3142444819487898, -0.31413843719245704, -0.31403556500089582,
-0.3139357519610217, -0.3138388900033446, -0.3137448761044012,
-0.3136536119031339, -0.3135650033970719, -0.31347896065874054,
-0.313395397577522, -0.3133142316243978, -0.3132353836372618,
-0.3131587776247182, -0.31308434058648726, -0.3130120023487362,
-0.3129416954128236, -0.312873354816102, -0.31280691800356786,
-0.3127423247092746, -0.3126795168465325, -0.3126184384060433,
-0.3125590353611786, -0.31250125557972824, -0.3124450487414865,
-0.31239036626114686, -0.3123371612159933, -0.3122853882779676,
-0.3122350036497119, -0.3121859650042474, -0.3121382314279689,
-0.312091763366685, -0.3120465225744507, -0.31200247206497295,
-0.31195957606538566, -0.31191779997222696, -0.3118771103094398,
-0.31183747468827366, -0.31179886176893457, -0.3117612412238927,
-0.3117245837027161, -0.31168886079835223, -0.31165404501476557,
-0.3116201097358512, -0.31158702919555464, -0.31155477844912954,
-0.3115233333454816, -0.3114926705005294, -0.31146276727155037,
-0.31143360173244494, -0.3114051526498949, -0.3113773994603619,
-0.3113503222478989, -0.311323901722733, -0.3112981192005923,
-0.31127295658274245, -0.31124839633671136, -0.3112244214776655,
-0.3112010155504241, -0.3111781626120766, -0.3111558472151897,
-0.31113405439157726, -0.311112769636615, -0.3110919788940804,
-0.311071668541499, -0.3110518253759788, -0.31103243660051827,
-0.31101348981077237, -0.31099497298225065, -0.31097687445795386,
-0.31095918293640706, -0.3109418874601061, -0.3109249774043339,
-0.31090844246635696, -0.31089227265498004, -0.3108764582804418,
-0.31086098994465666, -0.31084585853177255, -0.31083105519904464,
-0.31081657136801477, -0.31080239871598325, -0.31078852916776445,
-0.3107749548877154, -0.3107616682720315, -0.3107486619412973,
-0.310735928733289, -0.31072346169601117, -0.3107112540809679,
-0.3106992993366611, -0.3106875911022997, -0.3106761232017224,
-0.31066488963752287, -0.3106538845853671, -0.31064310238850557,
-0.31063253755246156, -0.31062218473990305, -0.3106120387656786,
-0.3106020945920245, -0.31059234732392826, -0.3105827922046445,
-0.3105734246113628, -0.3105642400510203, -0.31055523415624847,
-0.31054640268145983, -0.3105377414990593, -0.31052924659578574,
-0.3105209140691715, -0.3105127401241193, -0.3105047210695927,

-0.3104968533154193, -0.3104891333691902, -0.31048155783327147,
-0.3104741234019089, -0.3104668268584282, -0.31045966507252853,
-0.31045263499766307, -0.31044573366850803, -0.3104389581985099,
-0.3104323057775186, -0.31042577366949387, -0.3104193592102844,
-0.31041305980548733, -0.31040687292836827, -0.310400796117854,
-0.31039482697658777, -0.3103889631690495, -0.3103832024197347,
-0.31037754251139255, -0.3103719812833227, -0.3103665166297214,
-0.31036114649808794, -0.3103558688876776, -0.31035068184800063,
-0.3103455834773792, -0.3103405719215384, -0.31033564537225294,
-0.3103308020660252, -0.31032604028281824, -0.3103213583448151,
-0.3103167546152284, -0.31031222749714044, -0.3103077754323836,
-0.3103033969004513, -0.31029909041744813, -0.3102948545350699,
-0.31029068783961455, -0.3102865889510251, -0.3102825565219626,
-0.3102785892369052, -0.3102746858112801, -0.310270844990613,
-0.3102670655497151, -0.31026334629188623, -0.31025968604814375,
-0.31025608367647944, -0.31025253806113273, -0.31024904811189,
-0.31024561276340445, -0.3102422309745347, -0.3102389017277037,
-0.310235624028279, -0.3102323969039693, -0.31022921940423753,
-0.31022609059973755, -0.3102230095817572, -0.31021997546169,
-0.3102169873705118, -0.31021404445827894, -0.31021114589363935,
-0.31020829086335683, -0.3102054785718503, -0.3102027082407487,
-0.31019997910845154, -0.3101972904297116, -0.31019464147522097,
-0.31019203153121605, -0.3101894598990883, -0.3101869258950096,
-0.31018442884956626, -0.3101819681074054, -0.3101795430268867,
-0.31017715297975196, -0.31017479735079545, -0.3101724755375489,
-0.31017018694997317, -0.31016793101015816, -0.31016570715203423,
-0.31016351482108506, -0.310161353474076]