Identifying Pseudo-Concave Probabilistic Programs

Mila Lukic

Uvod

- Probabilistički programi specifiiraju složene probabilističke modele korišćenjem kompjuterskih programa.

```
1 x ~ uniform(0, 100)
```

x se uzorkuje iz uniformne distribucije.

```
2 b ~ normal(0.1*x+1, 1)
```

b se uzorkuje iz Gausove distribucije, gde je srednja vrednost funkcija od x.

```
3 observe (b, 6.29)
```

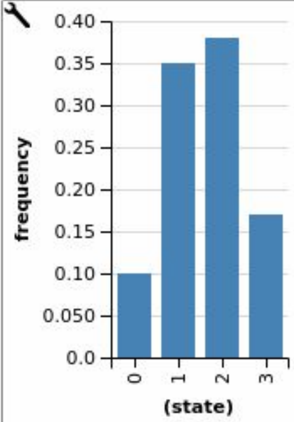
`observe` izjava, npr. `observe (b, 6.29)`, primenjuje Bajesovo pravilo da ažurira distribuciju.

Generativni Modeli

WebPPL - primeri (3 novcica)

```
var sumFlips = function() {
  return flip() + flip() + flip()
}
viz(repeat(100, sumFlips))
```

run

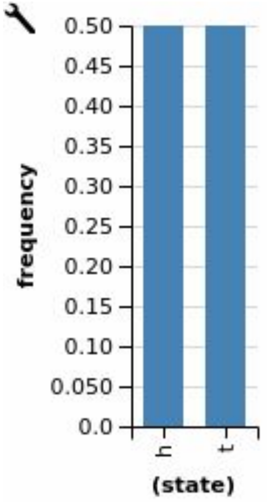


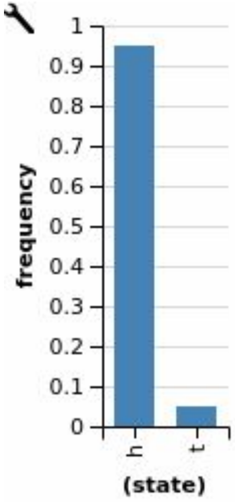
WebPPL - primeri (razliciti novcici)

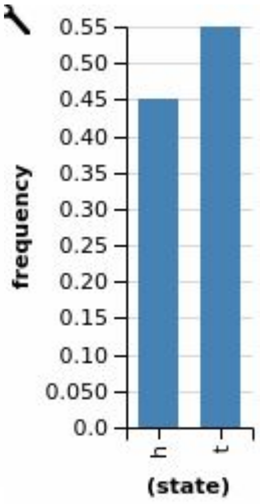
```
var makeCoin = function (weight) {
  return function() { return flip(weight) ? 'h' : 't' }
}

var fairCoin = makeCoin(0.5)
var trickCoin = makeCoin(0.95)
var bentCoin = makeCoin(0.25)

viz(repeat(20, fairCoin))
viz(repeat(20, trickCoin))
viz(repeat(20, bentCoin))
```







WebPPL - primeri (dijagnoze)

```
var lungCancer = flip(0.01)
var TB = flip(0.005)
var stomachFlu = flip(0.1)
var cold = flip(0.2)
var other = flip(0.1)

var cough =
    (cold && flip(0.5)) ||
    (lungCancer && flip(0.3)) ||
    (TB && flip(0.7)) ||
    (other && flip(0.01))

var fever =
    (cold && flip(0.3)) ||
    (stomachFlu && flip(0.5)) ||
    (TB && flip(0.1)) ||
    (other && flip(0.01))

var chestPain =
    (lungCancer && flip(0.5)) ||
    (TB && flip(0.5)) ||
    (other && flip(0.01))

var shortnessOfBreath =
    (lungCancer && flip(0.5)) ||
    (TB && flip(0.2)) ||
    (other && flip(0.01))

var symptoms = {
  cough: cough,
  fever: fever,
  chestPain: chestPain,
  shortnessOfBreath: shortnessOfBreath
}

symptoms
```

run

```
{"cough":false,"fever":false,"chestPain":false,"shortnessOfBreath":false}
```

AURA

AURA alat

- AURA je sistem za automatsku inferenciju u probabilističkim programima koristeći kvantizovano rezonovanje.
- Ovaj alat procenjuje granice posteriorne distribucije probabilistickih programa.

Umesto tacne vrednosti verovatnoce nekog ishoda P, imacemo interval [p1, p2]

Verovatnoca razlicitih ishoda nakon sto uzmemo u obzir nove dokaze/podatke.

- Tokom rada na ovom projektu, bavila sam se neprekidnim funkcijama.

Rad AURA alata sa “necistocama” u podacima

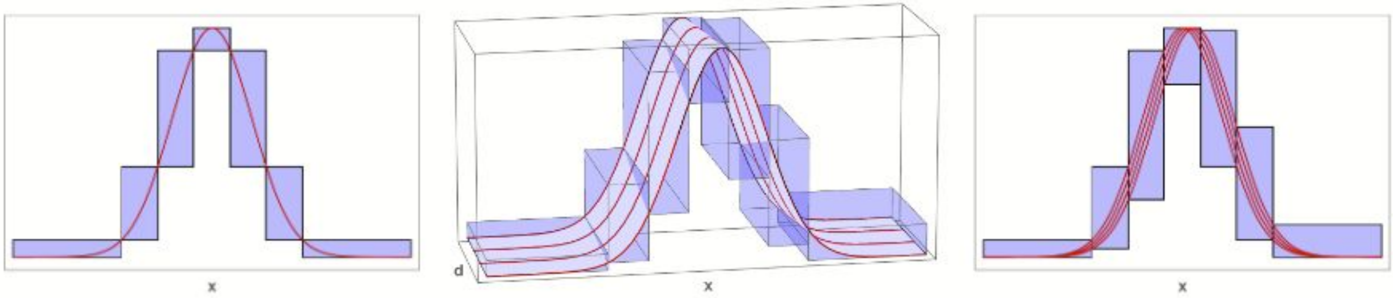


Fig. 2. Bounding Single Posterior Fig. 3. Data Perturbation Analysis Fig. 4. Bounding Data Perturb.

Rezultati: AURA je znacajno preciznija od naivnih alata za analizu intervala.

Nacin rada AURA alata

Primer: Bacamo 2 novcica, zelimo da posmatramo slucajeve u kojima nisu oba "glave"

benchmarks > psense_bench > coins > coins.template

```
float c1
float c2
c1 = flip(1/2)
c2 = flip(1/2)
float bothHeads
bothHeads = c1 && c2
hardObserve(bothHeads == 0)
```

AURA ce automatski generisati ekvivalentan python kod, koji pri pokretanju prikazuje raspodelu ove funkcije.

Tacnost AURA alata

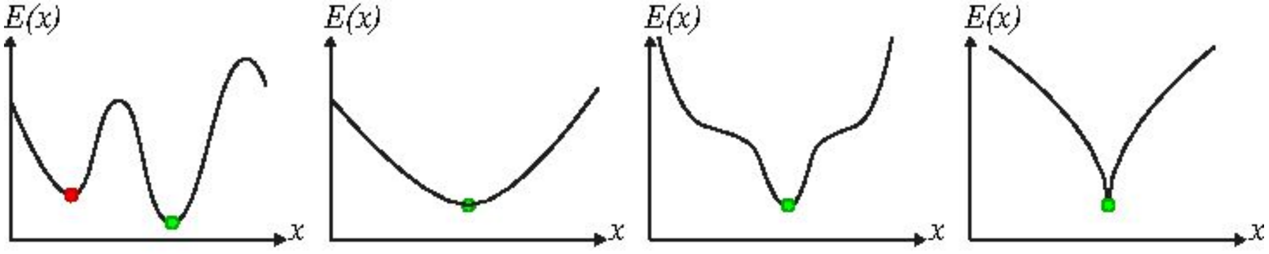
- Ukoliko su raspodele nekog probabilistickog programa **pseudo-konkavne,** racunanje intervala [p1, p2] za tu funkciju u AURA alatu bice sigurno tacno.*

- Postoje tehnike za automatsko testiranje da li je data funkcija pseudo-konveksna, odnosno pseudo-konkavna.
- Tehnika koju sam ja koristila bila je “**interval computation**”.

- **Tokom izrade projekta, saznala sam da je za mnoge funkcije dovoljno dokazati i da je njihov logaritam pseudo-konveksan/pseudo-konkavan.*

Pseudo-konveksne i pseudo-konkavne funkcije

- Pseudo-konveksne funkcije su funkcije koje se ponasaju kao konveksne kada je u pitanju pronalazenje njihovih minimuma.
- Ukoliko je funkcija f(x) pseudo-konveksna, onda je funkcija -f(x) pseudo-konkavna.



(a) Non-convex (b) Convex (c) Pseudoconvex (d) Quasiconvex

- **Mnoge standardne funkcije raspodele (Gausova, Beta, Gama...) su pseudo-konkavne.**

Algoritam za direktnu evaluaciju intervala

Algorithm 1 Direct interval evaluation

1. compute the interval matrix $\boldsymbol{M}^{\alpha}$ by (2),
2. **if** $\lambda_n(M_c^{\alpha}) \geq \rho(M_{\Delta}^{\alpha})$ **then**
3. $\quad f(x)$ is pseudoconvex on $\boldsymbol{x}$,
4. **else if** the matrix (3) is positive semidefinite for every $z \in \{\pm 1\}^{n-1} \times \{1\}$ **then**
5. $\quad f(x)$ is pseudoconvex on $\boldsymbol{x}$,
6. **else**
7. $\quad$ pseudoconvexity is not verified.
8. **end if**

Intervali

- Interval [a, b] predstavlja raspon izmedju dve vrednosti, gde *a* predstavlja donju, a *b* gornju granicu tog raspona.
- *Na primer: ako je t(g) vreme za koje je objektu potrebno da dodje do poda kad je bacen sa visine h=50*

$$t(g) = \frac{\sqrt{2h}}{g} = \frac{10}{\sqrt{g}}$$

"g" predstavlja gravitaciono ubrzanje, koje nike identicno na svakom mestu u svetu.

Najniže: Peru, $g = 9.76\,\mathrm{m/s^2}$

Najviše: Arktik, $g = 9.83\,\mathrm{m/s^2}$

$\longrightarrow$ Vrednost t(g) pripada intervalu: [3.19, 3.20]

Intervali i aritmetika nad njima

- Addition:

$$[x_1, x_2] + [y_1, y_2] = [x_1 + y_1, x_2 + y_2]$$

- Subtraction:

$$[x_1, x_2] - [y_1, y_2] = [x_1 - y_2, x_2 - y_1]$$

- Multiplication:

$$[x_1, x_2] \cdot [y_1, y_2] = [\min\{x_1 y_1, x_1 y_2, x_2 y_1, x_2 y_2\}, \max\{x_1 y_1, x_1 y_2, x_2 y_1, x_2 y_2\}]$$

- Division:

$$\frac{[x_1, x_2]}{[y_1, y_2]} = [x_1, x_2] \cdot \frac{1}{[y_1, y_2]},$$

where

$$\frac{1}{[y_1, y_2]} = \left[\tfrac{1}{y_2}, \tfrac{1}{y_1}\right] \qquad \text{if } 0 \notin [y_1, y_2]$$

Problemi pri implementaciji

- **Da li je $X*X == X^2$?**
- Za $X = [-1, 2]$:
  - $\mathbf{X*X}$ = [-1. 2] * [-1, 2] = [min{1, -2, 1, 4}, max{1, -2, 1, 4}] = **[-2, 4]**
  - $\mathbf{X^2}$ **=** $[-1, 2]^2$ = **[0, 4] jer kvadrirani izraz ne moze biti negativan!**
    - Takodje ne moze biti [1, 4] jer je 0 izmedju -1 i 2

```python
def __pow__(self, exponent):
    result = Interval(self.start, self.end)
    if isinstance(exponent, int):
        # Handle cases where the interval includes negative numbers and the exponent is even
        if exponent%2 == 0:
            result.start = max(result.start, 0)
    elif isinstance(exponent, float):
        return Interval(max(0, result.start ** exponent), result.end ** exponent)
    for i in range(exponent-1):
        result *= Interval(self.start, self.end)
    return result

def __rpow__(self, base):
    if isinstance(base, (int, float)):  # Base is a number, interval is the exponent
        if base > 0:  # Check if base is positive
            lower_bound = base**self.start
            upper_bound = base**self.end
            return Interval(lower_bound, upper_bound)
        else:
            raise ValueError("Base must be positive for real interval exponentiation.")
    else:
        raise TypeError("Base must be a real number.")
```

Intervali - jos neki bitni pojmovi

- Za interval $X = [x1, x2]$:
  - Sredina intervala (Midpoint): $Xc = \frac{1}{2}(x1+x2)$
  - Radius intervala: $Xd = \frac{1}{2}(x2-x1)$

- Na primer, za interval $X = [3, 4]$:
  - $Xc = 3.5$
  - $Xd = 0.5$

Matrica Intervala

Ako imamo neku matricu intervala:

$$A = \begin{pmatrix} [1,2] & [3,4] \\ [5,6] & [7,8] \end{pmatrix}$$

To zapravo znaci da njoj pripada svaka matrica izmedju ove dve (svaki element je ogranicen):

$$L = \begin{pmatrix} 1 & 3 \\ 5 & 7 \end{pmatrix} \qquad U = \begin{pmatrix} 2 & 4 \\ 6 & 8 \end{pmatrix}$$

Matrica Intervala - Midpoint i Radius Matrice

- Sredina i Radius matrice intervala racunaju se tako sto izracunamo sredinu i radius svakog pojedinacnog elementa te matrice:

$$M = \begin{pmatrix} 1.5 & 3.5 \\ 5.5 & 7.5 \end{pmatrix} \qquad R = \begin{pmatrix} 0.5 & 0.5 \\ 0.5 & 0.5 \end{pmatrix}$$

- Kada je u pitanju aritmetika nad matricama intervala, koristi se klasicna aritmetika nad matricama u kombinaciji sa aritmetikom intervala.

Specijalne Matrice

- Matrica A je **Z-matrica** ukoliko $\forall a_{ij} \in A,\ a_{ij} \leq 0,\ i \neq j$
  - (Svi elementi van dijagonale nisu pozitivni)
- Matrica A je **M-matrica** ukoliko *je Z-matrica i* $\exists u,\ Au > 0$

Na primer: za matricu $A = \begin{bmatrix} 3 & -1 \\ -2 & 4 \end{bmatrix}$ *i vektor* $u = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$ *vazi:* $Au = \begin{bmatrix} 2 \\ 2 \end{bmatrix} > 0$

*Za ovu matricu kazemo da je **pozitivno polu-definitivna***

- **H-matrica** je generalizacija M-matrice koja ne zahteva da elementi na dijagonali ne budu pozitivni.

Gradijent funkcije i Hessian Matrica

- Gradijent funkcije je vektor koji sadrži parcijalne derivacije te funkcije po svim njenim promenljivama. Za funkciju $f(x_1, x_2, \dots, x_n)$ gradijent je:

$$\nabla f = \left(\frac{\partial f}{\partial x_1}, \frac{\partial f}{\partial x_2}, \dots, \frac{\partial f}{\partial x_n} \right)$$

- Hessian matrica je kvadratna matrica koja sadrži sve druge parcijalne derivacije drugog reda funkcije. Ona se koristi za analizu lokalne konveksnosti funkcije više promenljivih.

$$H(f) = \begin{bmatrix} \frac{\partial^2 f}{\partial x_1^2} & \frac{\partial^2 f}{\partial x_1 \partial x_2} & \cdots & \frac{\partial^2 f}{\partial x_1 \partial x_n} \\ \frac{\partial^2 f}{\partial x_2 \partial x_1} & \frac{\partial^2 f}{\partial x_2^2} & \cdots & \frac{\partial^2 f}{\partial x_2 \partial x_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial^2 f}{\partial x_n \partial x_1} & \frac{\partial^2 f}{\partial x_n \partial x_2} & \cdots & \frac{\partial^2 f}{\partial x_n^2} \end{bmatrix}$$

Implementacija algoritma

Formulacija problema

- $x \in \mathbf{R}^n$ je neprazan interval
- f: $\mathbf{R}^n \to \mathbf{R}$ je dvostruko diferencijabilna funkcija na intervalu koji sadrzi x.

Algorithm 1 Direct interval evaluation

1. compute the interval matrix $\boldsymbol{M}^{\alpha}$ by (2),

Ⓣ f(x) je pseudokonveksna na x ukoliko je $M_\alpha := H + \alpha g g^T$ pozitivna i polu-definitivna (M-matrica) za svako $H \in \mathbf{R}^{n \times n}$ i $g \in \mathbf{R}^n$, gde je $\alpha \geq 0$.

```python
def compute_hessian(f, symbols):
    # Compute the Hessian matrix of a function.
    return sp.hessian(f, symbols)

def compute_gradient(f, symbols):
    # Compute the gradient vector of a function.
    gradient = [sp.diff(f, symbol) for symbol in symbols]
    return sp.Matrix(gradient)

def make_gradient_matrix(g):
    g_matrix = g*g.T
    # Expand each element of the matrix
    # g_matrix_expanded = g_matrix.applyfunc(sp.expand)
    return g_matrix

def compute_M(H, g, intervals, symbols, alpha):
    # Compute the matrix M_alpha using Hessian H, gradient g, and scalar alpha.
    g_matrix = make_gradient_matrix(g)
    M = alpha * g_matrix + H

    M_eval = [evaluate_expression(str(expr), intervals, symbols) for expr in M]
    return IntervalMatrix(M_eval)
```

Dovoljni uslovi pseudo-konveksnosti

. **if** $\lambda_n(M_c^\alpha) \geq \rho(M_\Delta^\alpha)$ **then**

. $f(x)$ is pseudoconvex on $\boldsymbol{x}$,

- M_c predstavlja Midpoint matricu M_α, dok M_Δ predstavlja njenu Radius matricu.
- Ovaj uslov proverava da li je **najmanja sopstvena vrednost M_c** veca ili jednaka **najvecoj sopstvenoj vrednosti M_Δ.**

```
def first_condition(M):
    # Check if the smallest eigenvalue of M_alpha_c is at least the largest eigenvalue of M_alpha_delta.
    min_evc = min(np.linalg.eigvals(M.calculateMidpoint()))
    max_evd = max(np.linalg.eigvals(M.calculateRadius()))
    return min_evc >= max_evd
```

Dovoljni uslovi pseudo-konveksnosti

4. **else if** the matrix (3) is positive semidefinite for every $z \in \{\pm 1\}^{n-1} \times \{1\}$ **then**
5. $f(x)$ is pseudoconvex on $\boldsymbol{x}$,

Proveravamo da li su sve sopstvene vrednosti ove kombinovane matrice pozitivne.

```
def second_condition(M):
    # Check if M_alpha_c - M_alpha_delta * diag(z) is positive semidefinite for every z.
    Mc = M.calculateMidpoint()
    Md = M.calculateRadius()
    z = np.diag([1] * M.rows)

    M_combined = Mc - np.dot(Md, z)

    # Check if all eigenvalues are non-negative
    return np.all(np.linalg.eigvals(M_combined) >= 0)
```

Primer rada algoritma

Primer pseudo-konveksne funkcije:

```
mila@sushi:~/Desktop/uiuc/pseudoconcavity/pseudoconvexity_analysis$ python3 main.py
 The function x**2 + y**2  is pseudoconvex on  [(-1, 1), (-1, 1)]  for alpha =  0.05
 The function x**2 + y**2  is pseudoconvex on  [(-1, 1), (-1, 1)]  for alpha =  0.1
 The function x**2 + y**2  is pseudoconvex on  [(-1, 1), (-1, 1)]  for alpha =  0.15
 The function x**2 + y**2  is pseudoconvex on  [(-1, 1), (-1, 1)]  for alpha =  0.2
 The function x**2 + y**2  is pseudoconvex on  [(-1, 1), (-1, 1)]  for alpha =  0.25
 The function x**2 + y**2  is NOT pseudoconvex on  [(-1, 1), (-1, 1)]  for alpha =  0.3
 The function x**2 + y**2  is NOT pseudoconvex on  [(-1, 1), (-1, 1)]  for alpha =  0.35
 The function x**2 + y**2  is NOT pseudoconvex on  [(-1, 1), (-1, 1)]  for alpha =  0.4
 The function x**2 + y**2  is NOT pseudoconvex on  [(-1, 1), (-1, 1)]  for alpha =  0.45
 The function x**2 + y**2  is NOT pseudoconvex on  [(-1, 1), (-1, 1)]  for alpha =  0.5
 The function x**2 + y**2  is NOT pseudoconvex on  [(-1, 1), (-1, 1)]  for alpha =  0.55
 The function x**2 + y**2  is NOT pseudoconvex on  [(-1, 1), (-1, 1)]  for alpha =  0.6
 The function x**2 + y**2  is NOT pseudoconvex on  [(-1, 1), (-1, 1)]  for alpha =  0.65
 The function x**2 + y**2  is NOT pseudoconvex on  [(-1, 1), (-1, 1)]  for alpha =  0.7
 The function x**2 + y**2  is NOT pseudoconvex on  [(-1, 1), (-1, 1)]  for alpha =  0.75
 The function x**2 + y**2  is NOT pseudoconvex on  [(-1, 1), (-1, 1)]  for alpha =  0.8
 The function x**2 + y**2  is NOT pseudoconvex on  [(-1, 1), (-1, 1)]  for alpha =  0.85
 The function x**2 + y**2  is NOT pseudoconvex on  [(-1, 1), (-1, 1)]  for alpha =  0.9
 The function x**2 + y**2  is NOT pseudoconvex on  [(-1, 1), (-1, 1)]  for alpha =  0.95
mila@sushi:~/Desktop/uiuc/pseudoconcavity/pseudoconvexity_analysis$
```

Primer funkcije koja nije pseudo-konveksna:

```
mila@sushi:~/Desktop/uiuc/pseudoconcavity/pseudoconvexity_analysis$ python3 main.py
The function -x**2 + y  is NOT pseudoconvex on  [(-1, 1), (-1, 1)]  for alpha =  0.05
The function -x**2 + y  is NOT pseudoconvex on  [(-1, 1), (-1, 1)]  for alpha =  0.1
The function -x**2 + y  is NOT pseudoconvex on  [(-1, 1), (-1, 1)]  for alpha =  0.15
The function -x**2 + y  is NOT pseudoconvex on  [(-1, 1), (-1, 1)]  for alpha =  0.2
The function -x**2 + y  is NOT pseudoconvex on  [(-1, 1), (-1, 1)]  for alpha =  0.25
The function -x**2 + y  is NOT pseudoconvex on  [(-1, 1), (-1, 1)]  for alpha =  0.3
The function -x**2 + y  is NOT pseudoconvex on  [(-1, 1), (-1, 1)]  for alpha =  0.35
The function -x**2 + y  is NOT pseudoconvex on  [(-1, 1), (-1, 1)]  for alpha =  0.4
The function -x**2 + y  is NOT pseudoconvex on  [(-1, 1), (-1, 1)]  for alpha =  0.45
The function -x**2 + y  is NOT pseudoconvex on  [(-1, 1), (-1, 1)]  for alpha =  0.5
The function -x**2 + y  is NOT pseudoconvex on  [(-1, 1), (-1, 1)]  for alpha =  0.55
The function -x**2 + y  is NOT pseudoconvex on  [(-1, 1), (-1, 1)]  for alpha =  0.6
The function -x**2 + y  is NOT pseudoconvex on  [(-1, 1), (-1, 1)]  for alpha =  0.65
The function -x**2 + y  is NOT pseudoconvex on  [(-1, 1), (-1, 1)]  for alpha =  0.7
The function -x**2 + y  is NOT pseudoconvex on  [(-1, 1), (-1, 1)]  for alpha =  0.75
The function -x**2 + y  is NOT pseudoconvex on  [(-1, 1), (-1, 1)]  for alpha =  0.8
The function -x**2 + y  is NOT pseudoconvex on  [(-1, 1), (-1, 1)]  for alpha =  0.85
The function -x**2 + y  is NOT pseudoconvex on  [(-1, 1), (-1, 1)]  for alpha =  0.9
The function -x**2 + y  is NOT pseudoconvex on  [(-1, 1), (-1, 1)]  for alpha =  0.95
mila@sushi:~/Desktop/uiuc/pseudoconcavity/pseudoconvexity_analysis$
```

Normalna raspodela:

```
mila@sushi:~/Desktop/uiuc/pseudoconcavity/pseudoconvexity_analysis$ python3 main.py
(2.718281828459045, 7.38905609893065)
The function 0.398942280401433/2.71828182845905**(x**2/2)  is NOT pseudoconvex  on  [(-2, -1)]
The function 0.398942280401433/2.71828182845905**(x**2/2)  is NOT log-convex  on  [(-2, -1)]
The function 0.398942280401433/2.71828182845905**(x**2/2)  is NOT pseudoconcave  on  [(-2, -1)]
The function -log(0.398942280401433/2.71828182845905**(x**2/2))  is log-concave on  [(-2, -1)]  for alpha =  0.0
mila@sushi:~/Desktop/uiuc/pseudoconcavity/pseudoconvexity_analysis$ python3 main.py
(2.718281828459045, 7.38905609893065)
The function 0.398942280401433/2.71828182845905**(x**2/2)  is NOT pseudoconvex  on  [(-3, 0)]
The function 0.398942280401433/2.71828182845905**(x**2/2)  is NOT log-convex  on  [(-3, 0)]
The function -0.398942280401433/2.71828182845905**(x**2/2)  is pseudoconcave on  [(-3, 0)]  for alpha =  0.0
The function -log(0.398942280401433/2.71828182845905**(x**2/2))  is log-concave on  [(-3, 0)]  for alpha =  0.0
mila@sushi:~/Desktop/uiuc/pseudoconcavity/pseudoconvexity_analysis$
```

Hvala na paznji :)