Reconocimiento Estadístico de Patrones - Tarea 1

Rubén Pérez Palacios Lic. Computación Matemática, Profesor: Johan Van Horebeek

March 29, 2023

1 Ejercicios

1.1 Librerías

Importaremos todas las librerías que se usaran durante toda la tarea.

[ ]:
```
import sys
import numpy as np
import pandas as pd
from pprint import pprint
import itertools

from scipy.optimize import minimize

import plotly.graph_objects as go
import plotly.express as px
from plotly.subplots import make_subplots

from sklearn.decomposition import PCA
from sklearn import manifold
from sklearn_som.som import SOM
from sklearn.manifold import TSNE
```

1.2 K-Means

1.2.1 Problema

A continuación se muestra el resultado obtenido al aplicar `kmeans()` a 4 conjuntos de datos. También se aplicó algun otro algoritmo de agrupamiento basado en el uso de representantes de cada grupo (marcados por x). Identifica para cada conjunto de datos, cuál corresponde a `kmeans()`.

1.2.2 Solución

- A: Podemos notar que en A1 los puntos de azules (de la frontera) debería estar particionada en dos `clusters` que debería estar delimitados por la línea trasada puesto que en cada semiplano delimitado por esa linea sus puntos están más cercanos al respectivo `center` de ese mismo semiplano. Por lo tanto A1 no es K-Means, con lo que concluimos que A2 es K-Means
- B: Podemos notar como en B1 el punto señalado es más cercano al `center` rodeado por puntos verdes que al `center` rodeado por puntos rojos, por lo que este debería ser del mismo `cluster`

que el de los puntos verdes. Por lo tanto B1 no es K-Means, con lo que concluimos que B2 es K-Means.

- C: Podemos notar como en C1 el punto señalado es más cercano al `center` rodeado por puntos azules que al `center` rodeado por puntos verdes, por lo que este debería ser del mismo `cluster` que el de los puntos azules. Por lo tanto B1 no es K-Means, con lo que concluimos que B2 es K-Means.
- D: Podemos notar como en D2 el punto señalado es más cercano al `center` rodeado por puntos azules que al `center` rodeado por puntos rojos, por lo que este debería ser del mismo `cluster` que el de los puntos azules. Por lo tanto B1 no es K-Means, con lo que concluimos que D1 es K-Means.

1.3 K-Means vs Agglomerative

1.3.1 Problema

¿Cuál método es más robusto contra datos atípicos: k-medias o agrupamiento aglomerativo? ¿De qué depende?

1.3.2 Solución

Tomaremos como definición para un dato que es atípico si es muy distinito a los demás, que en terminos de distancia esto es que su distancia hacia el resto de los datos sea muy grande con respecto al resto de las distnacias.

En el `K-Means` cuando tenemos datos atípicos, estos cuando sean agrupados con otros datos en sus respectivos grupos en el cálculo de sus `centers` se verán fuertemente influenciados por estos ya que la distancia es muy grande al resto de los puntos (con respecto al resto de las distancias) y con ello atraer con mayor facilidad a los `centers`. En cambio en el agrupamiento `Agglomerative` esto no pasara puesto que este agrupara a los datos atípicos al final puesto que su distancia es muy grande con respecto al resto de las distancias, con lo cuál estos influenciaran poco en el agrupamiento de los cercanos.

1.4 Medida de simlitud entre conjuntos

1.4.1 Problema

Sea S un conjunto finito. Definimos como medida de similitud entre dos subconjuntos A y B de S:

$$K(A, B) := |A \cap B|$$

Busca una función $\phi()$ tal que:

$$K(A, B) = \langle \Phi(A), \Phi(B) \rangle$$

1.4.2 Solución

Puesto que S es un conjunto finito entonces podemos enumerar sus elementos entonces sean $s_1, \cdots, s_d$ los elementos enumerados de S, donde $d = |S|$. Ahora sea \$

Sea $\Phi : 2^S \to R^d$ dada por

$$\phi(A)^i = \begin{cases} 1 & \text{si } s_i \in A, \\ 0 & \text{si no.} \end{cases}.$$

Entonces podemos ver que

$$\begin{aligned}
\langle \Phi(A), \Phi(B) \rangle &= \sum_{i=1}^{d} \Phi(A)^i \Phi(B)^i \\
&= \sum_{i \in \{1,\cdots,d\} | s_i \in A \cup B} \Phi(A)^i \Phi(B)^i + \sum_{i \in \{1,\cdots,d\} | s_i \notin A \cup B} \Phi(A)^i \Phi(B)^i \\
&= \sum_{i \in \{1,\cdots,d\} | s_i \in A \cup B} 1 + \sum_{i \in \{1,\cdots,d\} | s_i \notin A \cup B} 0 \\
&= |A \cap B| \\
&= K(A, B)
\end{aligned}$$

1.5 Kernel PCA

1.5.1 Problema

To do…

1.5.2 Solución

Sea

$$\vec{1} = \begin{pmatrix} 1 \\ \vdots \\ 1 \end{pmatrix}, v \in R^N$$

entonces

$$\mathbb{C}\vec{1} = \left(\mathbb{I} - \frac{1}{n} \vec{1}\vec{1}^t \right) \vec{1} = \vec{1} - \frac{1}{n} \vec{1}\vec{1}^t \vec{1} = \vec{1} - \frac{1}{n} \vec{1} n = \vec{1} - \vec{1} = 0,$$

por lo tanto

$$\mathbb{K}_{\mathbb{C}}^{\Phi} \vec{1} = \mathbb{C} K^{\Phi} \mathbb{C} \vec{1} = 0,$$

con lo que concluimos que $\vec{1}$ es vector propio de $\mathbb{K}_{\mathbb{C}}^{\Phi}$ con valor propio 0.

Ahora como

$$\mathbb{K}_{\mathbb{C}}^{\Phi}$$

es simétrica

$$\vec{1}^t \mathbb{K}_{\mathbb{C}}^{\Phi} = 0,$$

por lo tanto

$$\vec{1}\vec{1}^t\lambda_v v = \vec{1}\vec{1}^t\mathbb{K}_{\mathbb{C}}^{\Phi}v = \vec{1}0v = \vec{0},$$

debido a que $v \neq \vec{1}$ entonces $\lambda_v \neq 0$, por lo tanto $\vec{1}\vec{1}^t\lambda_v v = \vec{0}$ ssi $\vec{1}\vec{1}^t v = \vec{0}$,con lo cual concluimos que

$$\sum_i v_i = 0.$$

[ ]:
```
class Kernel_PCA:

    def __init__(self, data, sigma, dim):
        X = data.to_numpy()
        ii , jj = np.mgrid[0:X.shape[0],0:X.shape[0]]
        xx = X[ii]
        yy = X[jj]
        K_phi = np.exp(-np.power(np.linalg.norm(xx-yy, axis=2).squeeze(),2)/
 ↪sigma)
        C = np.identity(K_phi.shape[0]) - (1/K_phi.shape[0])*np.ones(K_phi.
 ↪shape)
        self.K_phi_c = C@K_phi@C
        self.eigen_values, self.eigen_vectors = np.linalg.eigh(self.K_phi_c)
        self.eigen_vectors = self.eigen_vectors[:,::-1]
        self.eigen_values = self.eigen_values[::-1]
        self.proj = self.eigen_vectors[:,:dim]
        return
```

[ ]:
```
colors_by_index = np.array(["#1F77B4", "#FF7F0E", "#2CA02C", "#D62728",␣
 ↪"#9467BD", "#8C564B", "#E377C2", "#7F7F7F", "#BCBD22", "#17BECF"])
index_of_colors = dict({
    "#1F77B4" : 0,
    "#FF7F0E" : 1,
    "#2CA02C" : 2,
    "#D62728" : 3,
    "#9467BD" : 4,
    "#8C564B" : 5,
    "#E377C2" : 6,
    "#7F7F7F" : 7,
    "#BCBD22" : 8,
    "#17BECF" : 9
})
```

[ ]:
```
def graph_point_sets(point_sets, title_sets, title, sigma):
    global colors_by_index
    fig = go.Figure()
    for (i, points), points_title in zip(enumerate(point_sets), title_sets):
```

```
        fig.add_scatter(
            x = points[:,0],
            y = points[:,1],
            name = points_title,
            marker_color = colors_by_index[i],
            mode="markers"
        )
    fig.update_layout(
        template="simple_white",
        title=title,
        width=1000,
        height=1000
    )
    fig.update_yaxes(scaleanchor="x",scaleratio=1)
    fig.show()
```

[ ]:
```
def sigma_animation(point_sets, sigma, dim, titles, title):
    points = (np.concatenate(point_sets))
    X = pd.DataFrame(points)
    proj = []
    for s in sigma:
        proj_sigma = Kernel_PCA(X, s, dim).proj
        proj += [proj_sigma]
    proj = np.array(proj)
    data = pd.DataFrame(
        {
            "x" : proj[:,:,0].flatten(),
             "y" : (np.zeros(proj[:,:,0].flatten().size) if dim < 2 else proj[:
↪,:,1].flatten()),
            "sigma" : np.repeat(sigma, points.shape[0]),
            "color" : np.array(colors_by_index[[np.
↪repeat(range(len(point_sets)),[points.shape[0] for points in␣
↪point_sets])]*len(sigma)]).flatten()
        }
    )
    fig = px.scatter(data, x="x",y="y",animation_frame="sigma", color="color")
    fig.update_layout(
        template="simple_white",
        title=title,
        width=1000,
        height=1000,
        sliders=[{"transition":dict(duration=0.01),'currentvalue':{"prefix":␣
↪r"\sigma = "},}]
    )
    tmin = data['x'].min()
    tmax = data['x'].max()
    fig.update_xaxes(
```

```
        showgrid=False,
        showline=True,
        zeroline=(True if dim == 1 else False),
        range=[tmin*(1-(-1 if tmin < 0 else 1)*.1), tmax*(1+(-1 if tmax < 0␣
↪else 1)*.1)]
    )
    tmin = data['y'].min()
    tmax = data['y'].max()
    fig.update_yaxes(
        showgrid=False,
        showline=True,
        zeroline=False,
        range=[tmin*(1-(-1 if tmin < 0 else 1)*.1), tmax*(1+(-1 if tmax < 0␣
↪else 1)*.1)],
    )
    fig.for_each_trace(lambda t: t.update(name = titles[index_of_colors[t.
↪name]],
                                          legendgroup = titles[index_of_colors[t.
↪name]],
                                          hovertemplate = t.hovertemplate.replace(t.
↪name, titles[index_of_colors[t.name]])
                                         )
                  )
    fig.show()
```

[ ]:
```
def graph_pca(point_sets, titles, title):
    pca = PCA(1)
    proj = pca.fit_transform(np.concatenate(point_sets)).flatten()
    fig = px.scatter(x=proj,y=np.zeros(proj.size), color=colors_by_index[np.
↪repeat(range(len(point_sets)),[points.shape[0] for points in point_sets])])
    fig.update_layout(
        template="simple_white",
        title=title,
        width=1000,
        height=1000
    )
    fig.update_xaxes(
        showgrid=False,
        showline=True,
        zeroline=True
    )
    fig.update_yaxes(
        showgrid=False,
        showline=True,
        zeroline=False,
        scaleanchor = "x",
        scaleratio = 1
```

```
    )
    fig.for_each_trace(lambda t: t.update(name = titles[index_of_colors[t.
↪name]],
                                          legendgroup = titles[index_of_colors[t.
↪name]],
                                          hovertemplate = t.hovertemplate.replace(t.
↪name, titles[index_of_colors[t.name]])
                                         )
                  )
    fig.show()
```

[ ]:
```
def solve(point_sets, title_sets, title, sigma, dim):
    graph_point_sets(point_sets, title_sets, title, sigma)
    graph_pca(point_sets, title_sets, title + "PCA")
    for d in range(1,dim+1):
        sigma_animation(point_sets, sigma, d, title_sets, title + ", KPCA : " +␣
↪str(d) + " - dim")
    return
```

[ ]:
```
sigma = [0.01,0.1,1,10,100,1000,10000]
p1 = np.random.multivariate_normal([-5,-5],np.identity(2),100)
p2 = np.random.multivariate_normal([5,5],np.identity(2),100)
solve([p1,p2], ["N((-5,-5),I)","N((5,5),I)"], "Normales Bivariadas", sigma, 2)
```

Pemos ver como en ambos casos tanto en el unidimensional como en el bidimensional se obtienen conjunto linealmente separabales con el `KPCA` cuando σ es grande. Así cómo mientras σ aumenta la distribución de nuestros puntos cada vez se parece más a la del `PCA`. Por último cuando σ es grande podemos ver como nuestro conjunto de puntos es separado por nuestra línea $x = 0$ con lo que corroboramos que agrupa nuestros datos.

[ ]:
```
sigma = [0.01,0.1,1,10,100,1000,10000]
p1 = np.array([np.array([0,0]) + np.array([np.cos(theta),np.sin(theta)]) for␣
↪theta in np.linspace(0,np.pi,50)])
p2 = np.array([np.array([1,0.5]) + np.array([np.cos(-theta),np.sin(-theta)])␣
↪for theta in np.linspace(0,np.pi,50)])
solve([p1,p2], ["C_1(0,0)","C_1(1,0.5)"], "Medio circulos", sigma, 2)
```

Pemos ver como en ambos casos tanto en el unidimensional como en el bidimensional se obtienen conjunto linealmente separabales con el `KPCA`, solo que ahora se obtiene en el caso unidimensional cuando σ es pequeño. Así cómo mientras σ aumenta la distribución de nuestros puntos cada vez se parece más a la del `PCA` (puedes hacer zoom a la gráfica seleccionando con el mouse una región). Por último cuando σ es pequeño podemos ver como nuestro conjunto de puntos es separado por nuestra línea $x = 0$ con lo que corroboramos que agrupa nuestros datos.

1.6 Aguacates

1.6.1 Problemas

Considera el archivo aguacate.xls con el precio promedio mensual del aguacate en 52 ciudades mexicanas en 2022. Es parte de los datos que el INEGI usa como insumo para el cálculo del Indice Nacional de Precios al Consumidor (INPC).

Encuentra algunas visualizaciones informativas (vistas en la primera parte del curso). ¿Encuentras evidencia de clusters de ciudades? Motiva e ilustra tu respuesta.

1.6.2 Solución

1.7 ME

1.7.1 Problema

Considera un torneo deportivo entre n equipos. Las reglas son tal que en cada partido un equipo gana o pierde, no hay empates. Supongamos que se llevaron a cabo cierto número de partidos. Sea $n_{i,j}$ el número de veces que equipo i ganó de equipo j (puede ser que nunca jugaron, entonces $n_{i,j} = n_{j,i} = 0$).

Un modelo sencillo para pomdelos los resultados de los partidos consiste en suponer que cada equipo i tiene un parámetro $\theta_i \in R^r$ asociado que refleja habilidad (fuerza) en el juego. El modelo supone que

$$P(\text{equipo } i \text{ gana de equipo } j) = \frac{\theta_i}{\theta_i + \theta_j}$$

Suponiendo independencia, y θ el vector con todas las θ_i, la verosimilitud es

$$L(\theta) = \prod_{i \neq j} \left(\frac{\theta_i}{\theta_i + \theta_j} \right)^{n_{i,j}}.$$

La logverosimilitud es

$$l(\theta) = \sum_{i \neq j} n_{i,j} \left(\ln(\theta_i) - \ln(\theta_i + \theta_j) \right).$$

Como $-ln()$ es convexa, se puede demostrar (no hay que hacerlo):

$$-lny \geq \ln x - \frac{y}{x} + 1.$$

Usando lo anterior para convencerte que la siguiente función $g\left(\theta|\theta^k\right)$ minoriza la log verosimilitud.

$$g\left(\theta|\theta^k\right) = \sum_{i \neq j} n_{i,j} \left(\ln(\theta_i) - \ln(\theta_i^k + \theta_j^k) - \frac{\theta_i + \theta_j}{\theta_i^k + \theta_j^k} + 1 \right)$$

Verifica que el máximo es

$$\theta_i^{k+1} = \frac{\sum_{i \neq j} n_{i,j}}{\sum_{i \neq j} \frac{n_{i,j}+n_{j,i}}{\theta_i^k + \theta_j^k}}$$

Implempéntalo para un ejemplo sencillo.

1.7.2 Solución

Primero notemos que por ser $-ln()$ convexa entonces

$$-\ln(\theta_i + \theta_j) \geq -\ln(\theta_i^k + \theta_j^k) - \frac{\theta_i + \theta_j}{\theta_i^k + \theta_j^k} + 1,$$

por lo que

$$l(\theta) \geq g\left(\theta | \theta^k\right).$$

Luego veamos que

$$\begin{aligned} g\left(\theta^k | \theta^k\right) &= \sum_{i \neq j} n_{i,j} \left(\ln(\theta_i^k) - \ln(\theta_i^k + \theta_j^k) - \frac{\theta_i^k + \theta_j^k}{\theta_i^k + \theta_j^k} + 1 \right) \\ &= \sum_{i \neq j} n_{i,j} \left(\ln(\theta_i^k) - \ln(\theta_i^k + \theta_j^k) \right) \\ &= l\left(\theta^k\right) \end{aligned}$$

.

Por lo tanto concluimos que g minoriza a f.

Por otra parte notemos que

$$\begin{aligned} \{i \neq j\} &= \{i \neq j\} \cap (\{i = l\} \cup \{i \neq l\}) \\ &= (\{i \neq j\} \cap \{i = l\}) \cup (\{i \neq j\} \cap \{i \neq l\} \cap (\{j = l\} \cup \{j \neq l\})) \\ &= (\{i \neq j\} \cap \{i = l\}) \cup (\{i \neq j\} \cap \{i \neq l\} \cap \{j = l\}) \cup (\{i \neq j\} \cap \{i \neq l\} \cap \{j \neq l\}) \end{aligned}$$

Entonces veamos que la derivada de g es

$$
\begin{aligned}
\frac{\partial g}{\partial \theta_l} &= \frac{\partial}{\partial \theta_l} \sum_{i \neq j} n_{i,j} \left(\ln(\theta_i) - \ln(\theta_i^k + \theta_j^k) - \frac{\theta_i + \theta_j}{\theta_i^k + \theta_j^k} + 1 \right) \\
&= \frac{\partial}{\partial \theta_l} \sum_{i \neq j \wedge i = l} n_{i,j} \left(\ln(\theta_i) - \ln(\theta_i^k + \theta_j^k) - \frac{\theta_i + \theta_j}{\theta_i^k + \theta_j^k} + 1 \right) \\
&+ \frac{\partial}{\partial \theta_l} \sum_{i \neq j \wedge i \neq l \wedge j = l} n_{i,j} \left(\ln(\theta_i) - \ln(\theta_i^k + \theta_j^k) - \frac{\theta_i + \theta_j}{\theta_i^k + \theta_j^k} + 1 \right) \\
&+ \frac{\partial}{\partial \theta_l} \sum_{i \neq j \wedge i \neq l \wedge j \neq l} n_{i,j} \left(\ln(\theta_i) - \ln(\theta_i^k + \theta_j^k) - \frac{\theta_i + \theta_j}{\theta_i^k + \theta_j^k} + 1 \right) \\
&= \frac{\partial}{\partial \theta_l} \sum_{i \neq j \wedge i = l} n_{i,j} \left(\ln(\theta_i) - \ln(\theta_i^k + \theta_j^k) - \frac{\theta_i + \theta_j}{\theta_i^k + \theta_j^k} + 1 \right) \\
&+ \frac{\partial}{\partial \theta_l} \sum_{i \neq j \wedge i \neq l \wedge j = l} n_{i,j} \left(\ln(\theta_i) - \ln(\theta_i^k + \theta_j^k) - \frac{\theta_i + \theta_j}{\theta_i^k + \theta_j^k} + 1 \right) \\
&= \frac{\partial}{\partial \theta_l} \sum_{j \neq l} n_{l,j} \left(\ln(\theta_l) - \ln(\theta_l^k + \theta_j^k) - \frac{\theta_l + \theta_j}{\theta_l^k + \theta_j^k} + 1 \right) \\
&+ \frac{\partial}{\partial \theta_l} \sum_{i \neq l} n_{i,l} \left(\ln(\theta_i) - \ln(\theta_i^k + \theta_l^k) - \frac{\theta_i + \theta_l}{\theta_i^k + \theta_l^k} + 1 \right) \\
&= \sum_{j \neq l} n_{l,j} \frac{\partial}{\partial \theta_l} \left(\ln(\theta_l) - \ln(\theta_l^k + \theta_j^k) - \frac{\theta_l + \theta_j}{\theta_l^k + \theta_j^k} + 1 \right) \\
&+ \sum_{i \neq l} n_{i,l} \frac{\partial}{\partial \theta_l} \left(\ln(\theta_i) - \ln(\theta_i^k + \theta_l^k) - \frac{\theta_i + \theta_l}{\theta_i^k + \theta_l^k} + 1 \right) \\
&= \sum_{j \neq l} n_{l,j} \left(\frac{1}{\theta_l} - \frac{1}{\theta_l^k + \theta_j^k} \right) \\
&+ \sum_{i \neq l} n_{i,l} \left(-\frac{1}{\theta_i^k + \theta_l^k} \right) \\
&= \sum_{i \neq l} n_{l,i} \left(\frac{1}{\theta_l} - \frac{1}{\theta_l^k + \theta_i^k} \right) \\
&+ \sum_{i \neq l} n_{i,l} \left(-\frac{1}{\theta_i^k + \theta_l^k} \right) \\
&= \sum_{i \neq l} n_{l,i} \left(\frac{1}{\theta_l} - \frac{1}{\theta_l^k + \theta_i^k} \right) + n_{i,l} \left(-\frac{1}{\theta_i^k + \theta_l^k} \right) \\
&= \frac{1}{\theta_l} \sum_{i \neq l} n_{l,i} - \sum_{i \neq l} \frac{n_{l,i} + n_{i,l}}{\theta_l^k + \theta_i^k}
\end{aligned}
$$

Veamos cuales son los puntos críticos de g

$$0 = \frac{1}{\theta_l}\sum_{i\neq l} n_{l,i} - \sum_{i\neq l}\frac{n_{l,i}+n_{i,l}}{\theta_l^k+\theta_i^k} \iff \theta_l = \frac{\sum_{i\neq l} n_{l,i}}{\sum_{i\neq l}\frac{n_{l,i}+n_{i,l}}{\theta_l^k+\theta_i^k}}$$

Además podemos ver que

$$\begin{aligned}
\frac{\partial^2 g}{\partial\theta_l^2} &= \frac{\partial}{\partial\theta_l}\left[\frac{1}{\theta_l}\sum_{i\neq l} n_{l,i} - \sum_{i\neq l}\frac{n_{l,i}+n_{i,l}}{\theta_l^k+\theta_i^k}\right] \\
&= \frac{\partial}{\partial\theta_l}\frac{1}{\theta_l}\sum_{i\neq l} n_{l,i} - \frac{\partial^2}{\partial\theta_l^2}\sum_{i\neq l}\frac{n_{l,i}+n_{i,l}}{\theta_l^k+\theta_i^k} \\
&= -\sum_{i\neq l} n_{l,i}\frac{1}{\theta_l^2} \\
&\leq 0
\end{aligned}$$

por lo tanto el punto críticos es máximo.

[ ]:
```
n = np.array(
    [
        [0,5],
        [3,0]
    ]
)

def l(theta):
    ii , jj = np.mgrid[0:theta.shape[0],0:theta.shape[0]]
    theta_x = theta[ii]
    theta_y = theta[jj]
    return np.sum(n*(np.log(theta_x)-np.log(theta_x+theta_y)))

def g(theta, theta_k):
    ii , jj = np.mgrid[0:theta.shape[0],0:theta.shape[0]]
    theta_x = theta[ii]
    theta_y = theta[jj]
    theta_k_x = theta_k[ii]
    theta_k_y = theta_k[jj]
    return np.sum(n*(np.log(theta_x)-np.
 ↪log(theta_k_x+theta_k_y)-((theta_x+theta_y)/(theta_k_x+theta_k_y))+1))

t=100
x = np.linspace(0.00001,1,t)
y = np.linspace(0.00001,1,t)
fig = go.Figure()
fig.add_surface(
    x = x,
    y = y,
```

```
    z = np.array([l(np.array([x_i,y_i])) for y_i, x_i in itertools.
 ↪product(y,x)]).reshape(t,t),
    name = 'l'
)
theta = np.array([1,0.1])
x = [theta]
bnds = ((0, None), (0, None))
for i in range(10):
    theta = minimize((lambda x : -g(x, theta)), theta, bounds=bnds).x
    print(theta)
    x += [theta]
x = np.array(x)
fig.add_trace(
    go.Scatter3d(
        x = x[1:,0],
        y = x[1:,1],
        z = np.array([g(theta_next, theta_act) for theta_next, theta_act in␣
 ↪zip(x[1:],x[:-1])]),
        name = "g(\theta_{k+1}|\theta_k)"
    )
)
fig.update_layout(
        template="simple_white",
        title="Torneo",
        width=1000,
        height=1000
)
fig.show()
```

```
[0.68749963 0.41249992]
[0.68749963 0.41249992]
[0.68749963 0.41249992]
[0.68749963 0.41249992]
[0.68749963 0.41249992]
[0.68749963 0.41249992]
[0.68749963 0.41249992]
[0.68749963 0.41249992]
[0.68749963 0.41249992]
[0.68749963 0.41249992]
```