Memo on Bregman balls

Basile Fraboni

Notations. All variables x, y, p, q, c used below are d-dimensional points in space and their individual components are denoted with subscripts, for instance x_i. Other notations such as τ are real scalar values.

Bregman divergence. In [Nielsen et al., 2009, intro] and [Nielsen, 2021a, eq. 35] we have the generic Bregman divergence equation that accepts any generator function F:

$$B_F(p, q) = F(p) - F(q) - (p - q)^T \nabla F(q) \tag{1}$$

KL generator. In [Nielsen et al., 2009, intro] we have the generator for the Kullback-Leibler divergence, the negative Shannon entropy:

$$F_{\text{KL}}(x) = \sum_i^d x_i \log(x_i) \tag{2}$$

Its gradient can be evaluated as follows:

$$\nabla F_{\text{KL}}(x) = \begin{bmatrix} \frac{\partial F_{\text{KL}}(x)}{\partial x_i} \\ \vdots \\ \frac{\partial F_{\text{KL}}(x)}{\partial x_d} \end{bmatrix} = \begin{bmatrix} \log(x_i) + 1 \\ \vdots \\ \log(x_d) + 1 \end{bmatrix} \tag{3}$$

Extended KL generator. In [Nielsen, 2021a, eq. 86] we have another generator for the Kullback-Leibler divergence, the extended negative Shannon entropy:

$$F_{\text{EKL}}(x) = \sum_i^d x_i \log(x_i) - x_i \tag{4}$$

Its gradient can be evaluated as follows:

$$\nabla F_{\text{EKL}}(x) = \begin{bmatrix} \frac{\partial F_{\text{EKL}}(x)}{\partial x_i} \\ \vdots \\ \frac{\partial F_{\text{EKL}}(x)}{\partial x_d} \end{bmatrix} = \begin{bmatrix} \log(x_i) \\ \vdots \\ \log(x_d) \end{bmatrix} \tag{5}$$

KL divergence. Both generators lead to the same dual Bregman divergence formula [Nielsen, 2021a, eq. 90]:

$$B_{\text{KL}}(x, y) = \sum_i^d x_i \log\left(\frac{x_i}{y_i}\right) + y_i - x_i \tag{6}$$

IS generator. In [Nielsen, 2021a, eq. 92] we have the generator for the Itakura-Saito divergence, the negative Burg entropy:

$$F_{\text{IS}}(x) = -\sum_i^d \log(x_i) \tag{7}$$

Its gradient can be evaluated as follows:

$$\nabla F_{\text{IS}}(x) = \begin{bmatrix} \frac{\partial F_{\text{IS}}(x)}{\partial x_i} \\ \vdots \\ \frac{\partial F_{\text{IS}}(x)}{\partial x_d} \end{bmatrix} = \begin{bmatrix} -\frac{1}{x_i} \\ \vdots \\ -\frac{1}{x_d} \end{bmatrix} \tag{8}$$

IS divergence. The negative Burg entropy leads to the Itakura-Saito divergence formula [Nielsen, 2021a, eq. 94]:

$$B_{\text{IS}}(x, y) = \sum_i^d \frac{x_i}{y_i} - \log\left(\frac{x_i}{y_i}\right) - 1 \tag{9}$$

Bregman balls. A Bregman ball of center c and of radius τ is defined as follows:

$$\text{ball}_{B_{\text{F}}}(c, \tau) = \{x : B_{\text{F}}(c, x) = \tau\} \tag{10}$$

A Bregman KL ball is then:

$$\text{ball}_{B_{\text{KL}}}(c, \tau) = \{x : B_{\text{KL}}(c, x) = \tau\} \tag{11}$$

And a Bregman IS ball:

$$\text{ball}_{B_{\text{IS}}}(c, \tau) = \{x : B_{\text{IS}}(c, x) = \tau\} \tag{12}$$

Note that the order of parameters in the divergence matters, as for example the KL divergence is not a symmetric function, $B_{\text{KL}}(x, y) \neq B_{\text{KL}}(y, x)$.

Implicit equation of Bregman balls. We can also define the equivalent implicit function as follows:

$$f(x, c, \tau) = B_{\text{F}}(c, x) - \tau \tag{13}$$

This implicit function can be drawn using contour plotting methods such as grid subdivision and bisection and solving $f(x, c, \tau) = 0$.

Parametric equations of Bregman balls. A more recent formulation proposed in [Nielsen, 2021a, sec. 2.5 and 2.6] allows to directly express a Bregman ball in a parametric form using the Lambert W function (specifically the real valued branches W_0 and W_{-1}). In 2D this lead to one equation for each quadrant.

$$\mathrm{ball}_{B_{\mathrm{KL}}}(c,\tau) = \begin{cases} x_{\mathrm{bl}}(u,c,\tau) = (x_1(u,c,\tau), y_1(u,c,\tau)) \\ x_{\mathrm{tl}}(u,c,\tau) = (x_1(u,c,\tau), y_2(u,c,\tau)) \\ x_{\mathrm{tr}}(u,c,\tau) = (x_2(u,c,\tau), y_2(u,c,\tau)) \\ x_{\mathrm{br}}(u,c,\tau) = (x_2(u,c,\tau), y_1(u,c,\tau)) \end{cases} \qquad u \in [0,\tau] \tag{14}$$

We detail next the parametric forms induced by the KL and the IS divergence. This approach can be generalized to arbitrary number of dimensions.

Parametric equations of Bregman KL balls. The parametric functions x_1, x_2, y_1 and y_2 are computed as follows for the KL divergence [Nielsen, 2021a, eq. 116-117]:

$$\begin{aligned} x_1(u,c,\tau) &= -c_x W_0\left(-\exp\left(-\frac{u}{c_x}-1\right)\right) \\ y_1(u,c,\tau) &= -c_y W_0\left(-\exp\left(-\frac{\tau-u}{c_y}-1\right)\right) \\ x_2(u,c,\tau) &= -c_x W_{-1}\left(-\exp\left(-\frac{u}{c_x}-1\right)\right) \\ y_2(u,c,\tau) &= -c_y W_{-1}\left(-\exp\left(-\frac{\tau-u}{c_y}-1\right)\right) \end{aligned} \tag{15}$$

Parametric equations of Bregman IS balls. The parametric functions x_1, x_2, y_1 and y_2 are computed as follows for the IS divergence [Nielsen, 2021a, eq. 118-119] :

$$\begin{aligned} x_1(u,c,\tau) &= -\frac{c_x}{W_{-1}(-\exp(-u-1))} \\ y_1(u,c,\tau) &= -\frac{c_y}{W_{-1}(-\exp(-(\tau-u)-1))} \\ x_2(u,c,\tau) &= -\frac{c_x}{W_0(-\exp(-u-1))} \\ y_2(u,c,\tau) &= -\frac{c_y}{W_0(-\exp(-(\tau-u)-1))} \end{aligned} \tag{16}$$

Visualization. We illustrate in Figure 1 and Figure 2 a visual comparison of the two approaches of implicit and parametric contour plotting. Note that the parametric version is much faster than the implicit plot. This allows fast visualization of Bregman vantage points tree as depicted in Figure 4. The python code to generates Figure 1 and Figure 2 can be found in Appendix A and Appendix B and the C++ code that generates the Bregman vantage points tree visualization can be found in the following repository `https://github.com/bfraboni/vptree-draw`.

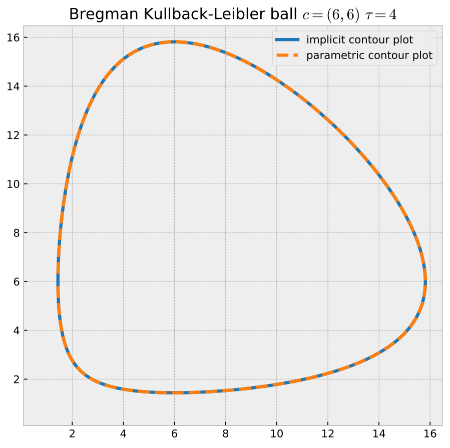

Figure 1: Comparison of implicit and parametric plots of a Bregman KL ball.

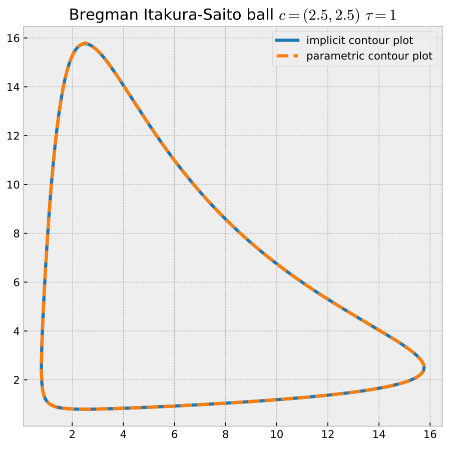

Figure 2: Comparison of implicit and parametric plots of a Bregman IS ball.

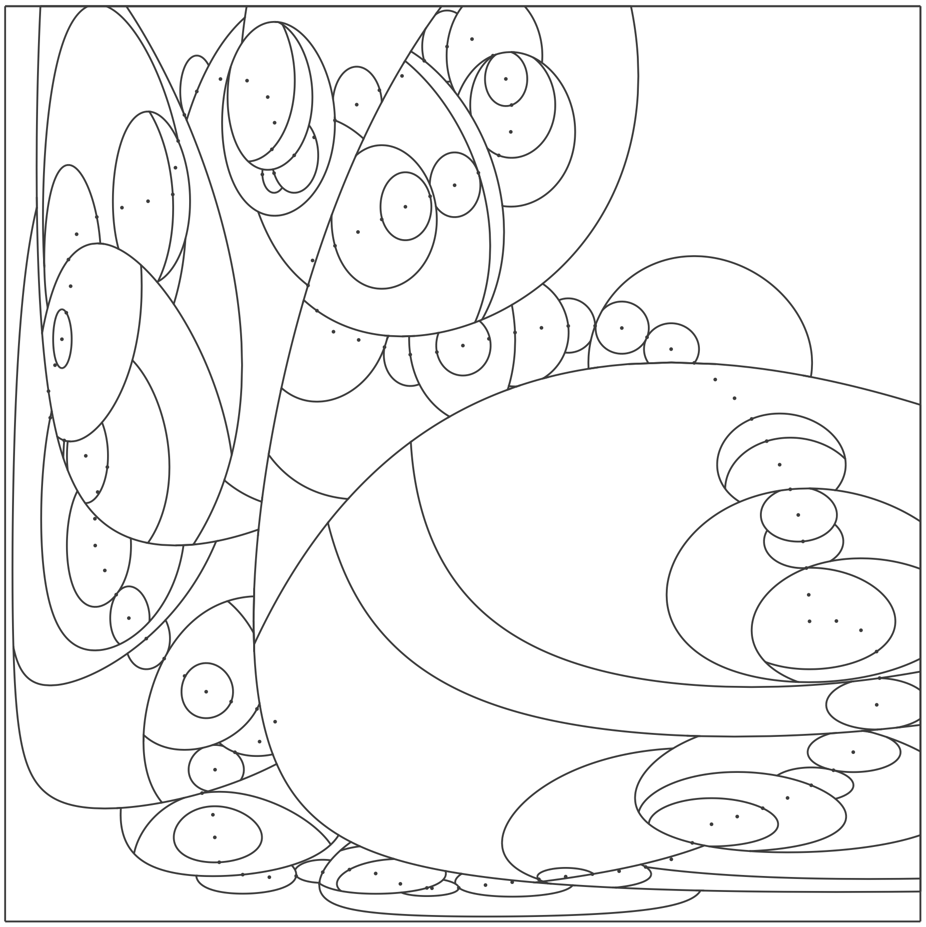

Figure 3: Visualization of a Bregman vantage points tree using parametric plots of Bregman KL cells.

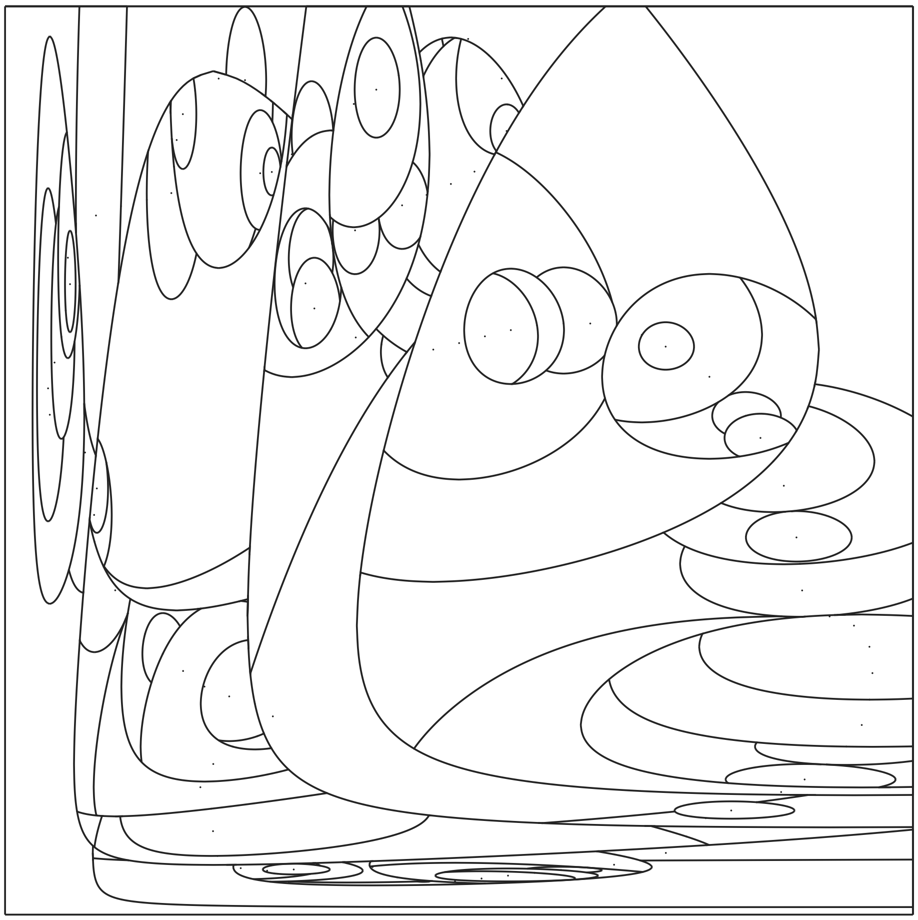

Figure 4: Visualization of a Bregman vantage points tree using parametric plots of Bregman IS cells.

References

F. Nielsen. On geodesic triangles with right angles in a dually flat space, 2021a. URL https://arxiv.org/abs/1910.03935.

F. Nielsen. On geodesic triangles with right angles in a dually flat space. In *Progress in Information Geometry*, pages 153–190. Springer, 2021b.

F. Nielsen, P. Piro, and M. Barlaud. Bregman vantage point trees for efficient nearest neighbor queries. In *2009 IEEE International Conference on Multimedia and Expo*, pages 878–881. IEEE, 2009.

A Python code to generate Figure 1

```python
import numpy as np
import matplotlib.pyplot as plt
from matplotlib.lines import Line2D
import scipy.special.lambertw as lambertw

# Parametric Bregman KL ball of center (cx,cy) and radius r
def klball(cx, cy, r, nb=250):
    x = np.zeros(4*nb)
    y = np.zeros(4*nb)
    du=r/(nb-1)
    # top left quadrant: (x1, y2)
    for i in np.arange(0, nb):
        u = r-(i+0.5) * du
        x[i]=-cx*np.real(lambertw(-np.exp(-u/cx-1), k=0))
        y[i]=-cy*np.real(lambertw(-np.exp(-(r-u)/cy-1), k=-1))
    # top right quadrant: (x2, y2)
    for i in np.arange(0, nb):
        u = (i+0.5) * du
        x[nb+i]=-cx*np.real(lambertw(-np.exp(-u/cx-1), k=-1))
        y[nb+i]=-cy*np.real(lambertw(-np.exp(-(r-u)/cy-1), k=-1))
    # bottom right quadrant: (x2, y1)
    for i in np.arange(0, nb):
        u = r-(i+0.5) * du
        x[2*nb+i]=-cx*np.real(lambertw(-np.exp(-u/cx-1), k=-1))
        y[2*nb+i]=-cy*np.real(lambertw(-np.exp(-(r-u)/cy-1), k=0))
    # bottom left quadrant: (x1, y1)
    for i in np.arange(0, nb):
        u = (i+0.5) * du
        x[3*nb+i]=-cx*np.real(lambertw(-np.exp(-u/cx-1), k=0))
        y[3*nb+i]=-cy*np.real(lambertw(-np.exp(-(r-u)/cy-1), k=0))

    return x,y

# Bregman ball center and radius
cx=6
cy=6
r=4

# implicit contour plot
delta = 0.025
xrange = np.arange(0.1, 16.5, delta)
yrange = np.arange(0.1, 16.5, delta)
X, Y = np.meshgrid(xrange,yrange)
# F is one side of the equation, G is the other
F = X-cx+cx*np.log(cx/X)
G = r-(Y-cy+cy*np.log(cy/Y))

# parametric contour plot
x, y = klball(cx, cy, r)

# figure
with plt.style.context("bmh"):
    plt.figure(figsize=(6,6))
    plt.contour(X, Y, (F - G), [0], colors=["tab:blue"], linewidths=[3])
    plt.plot(x, y, ls='--', lw=3, color= "tab:orange", label="parametric contour plot")
    handles, labels = plt.gca().get_legend_handles_labels()
    handles.insert(0, Line2D([0], [0], color='tab:blue', lw=3))
    labels.insert(0, 'implicit contour plot')
    plt.legend(handles, labels)
    plt.title(r'Bregman Kullback-Leibler ball $c=({},{})$ $\tau={}$'.format(cx,cy,r))
    plt.tight_layout()
    plt.savefig("klbball.pdf")
    plt.show()
```

B Python code to generate Figure 2

```python
import numpy as np
import matplotlib.pyplot as plt
from matplotlib.lines import Line2D
import scipy.special.lambertw as lambertw

# Parametric Bregman IS ball of center (cx,cy) and radius r
def isball(cx, cy, r, nb=250):
    x = np.zeros(4*nb)
    y = np.zeros(4*nb)
    du=r/(nb-1)
    # top left quadrant: (x1, y2)
    for i in np.arange(0, nb):
        u = r-(i+0.5) * du
        x[i]=-cx/np.real(lambertw(-np.exp(-u-1), k=-1))
        y[i]=-cy/np.real(lambertw(-np.exp(-(r-u)-1), k=0))
    # top right quadrant: (x2, y2)
    for i in np.arange(0, nb):
        u = (i+0.5) * du
        x[nb+i]=-cx/np.real(lambertw(-np.exp(-u-1), k=0))
        y[nb+i]=-cy/np.real(lambertw(-np.exp(-(r-u)-1), k=0))
    # bottom right quadrant: (x2, y1)
    for i in np.arange(0, nb):
        u = r-(i+0.5) * du
        x[2*nb+i]=-cx/np.real(lambertw(-np.exp(-u-1), k=0))
        y[2*nb+i]=-cy/np.real(lambertw(-np.exp(-(r-u)-1), k=-1))
    # bottom left quadrant: (x1, y1)
    for i in np.arange(0, nb):
        u = (i+0.5) * du
        x[3*nb+i]=-cx/np.real(lambertw(-np.exp(-u-1), k=-1))
        y[3*nb+i]=-cy/np.real(lambertw(-np.exp(-(r-u)-1), k=-1))

    return x,y

# Bregman ball center and radius
cx=2.5
cy=2.5
r=1

# implicit contour plot
delta = 0.025
xrange = np.arange(0.1, 16.5, delta)
yrange = np.arange(0.1, 16.5, delta)
X, Y = np.meshgrid(xrange,yrange)
# F is one side of the equation, G is the other
F = cx/X-np.log(cx/X)-1
G = r-(cy/Y-np.log(cy/Y)-1)

# parametric contour plot
x, y = isball(cx, cy, r)

# figure
with plt.style.context("bmh"):
    plt.figure(figsize=(6,6))
    plt.contour(X, Y, (F - G), [0], colors=["tab:blue"], linewidths=[3])
    plt.plot(x, y, ls='--', lw=3, color= "tab:orange", label="parametric contour plot")
    handles, labels = plt.gca().get_legend_handles_labels()
    handles.insert(0, Line2D([0], [0], color='tab:blue', lw=3))
    labels.insert(0, 'implicit contour plot')
    plt.legend(handles, labels)
    plt.title(r'Bregman Itakura-Saito ball $c=({},{})$ $\tau={}$'.format(cx,cy,r))
    plt.tight_layout()
    plt.savefig("isbball.pdf")
    plt.show()
```