# Lec 8 Ex.

1] **Beta Distribution:-**

$$\frac{\gamma(\alpha+\beta)}{\gamma(\alpha)\gamma(\beta)} x^{(\alpha-1)} (1-x)^{\beta-1}$$

(a) **Likelihood function:-**

$$L(\theta) = p(x_1, x_2 \dots x_n | \alpha, \beta)$$

$$= \prod_{i=1}^{n} \frac{\gamma(\alpha+\beta)}{\gamma(\alpha)\gamma(\beta)} x^{(\alpha-1)} (1-x)^{\beta-1}$$

$$\boxed{L(\theta) = \left[\frac{\gamma(\alpha+\beta)}{\gamma(\alpha)\gamma(\beta)}\right]^{n} \prod_{i=1}^{n} x^{(\alpha-1)n} \prod_{i=1} (1-x)^{n(\beta-1)}}$$

(b) **Log Likelihood:-**

$$l(\theta) = \log L(\theta)$$

PTO

(b)
$$\boxed{\begin{aligned} l(\theta) = {} & n \log \gamma(\alpha+\beta) - n \log \gamma(\alpha) - n \log \gamma(\beta) \\ & + \cancel{n}(\alpha-1)\sum_{i=1}^{n} \log x_i + \cancel{n}(\beta-1)\sum_{i=1}^{n} \log (1-x_i) \end{aligned}}$$

Negative log Likelihood:

$$\boxed{\begin{aligned} -l(\theta) = {} & -n \log \gamma(\alpha+\beta) + n \log \gamma(\alpha) + n \log \gamma(\beta) \\ & - \cancel{n}(\alpha-1)\sum_{i=1}^{n} \log x - \cancel{n}(\beta-1)\sum_{i=1}^{n} \log (1-x_i) \end{aligned}}$$

c] Computing partial Derivatives:

$$\cancel{\frac{dl}{d\alpha} = -n \frac{d}{d\alpha} \gamma(\alpha+\beta) + n \log}$$

$$\frac{dl}{d\alpha} = -n \frac{d \log(\gamma(\alpha+\beta))}{d(\gamma(\alpha+\beta))} + n \frac{d \log(\gamma(\alpha))}{d\alpha}$$

$$- \sum_{i=1}^{n} \log (x_i) = 0$$

$$\frac{\partial l}{\partial \beta} = -n \frac{\partial \log(\gamma(\alpha+\beta))}{\partial(\gamma(\alpha+\beta))} + n \frac{\partial \log(\gamma(\beta))}{\partial \beta}$$

$$- \sum_{i=1}^{n} \log(1-x_i) = 0$$

$$\boxed{\begin{aligned} \alpha_i &\leftarrow \alpha_{i-1} - \gamma \frac{\partial l}{\partial \alpha} \\[2em] \beta_i &\leftarrow \beta_{i-1} - \gamma \frac{\partial l}{\partial \beta} \end{aligned}}$$

(d) Gradient descent Algo:-

Step 1:- Take initial value of $\alpha$

Take initial value of $\beta$

Step 2:- Set max Iteration to $\approx$ 10,000

Step 3:- for $i = 1 : 10{,}000$

Step 4:- $\rightarrow \alpha_i = \alpha_{i-1} - \gamma \left.\frac{\partial l}{\partial \alpha}\right|_{\alpha}$

$\rightarrow \beta_i = \beta_{i-1} - \gamma \left.\frac{\partial l}{\partial \beta}\right|_{\beta}$

Step 5: $\cancel{(|\alpha_i - \alpha_{i-1}| < \varepsilon)}$

if $(|\alpha_i - \alpha_{i-1}| < \varepsilon$ && $|\beta_i - \beta_{i-1}| < \varepsilon)$

terminate ; end;

Step 6: end for.

In [1]:
```
using Distributions;
using Gadfly, Cairo, Fontconfig
```

In [2]:
```
function dl_by_da(samples, a, b)
    n = length(samples);
    result =  - n*digamma(a+b)+n*digamma(a)-sum(log.(samples));
    return result;
end
```

Out[2]: dl_by_da (generic function with 1 method)

In [3]:
```
function dl_by_db(samples, a, b)
    n = length(samples);
    result =  - n*digamma(a+b)+n*digamma(b)-sum(log.(1-samples));
    return result;
end
```

Out[3]: dl_by_db (generic function with 1 method)

In [4]:
```
function gradient_desent_beta(samples, learning_rate, max_iterations)
    n = length(samples);

    max_acc = 0.0001;

    a = rand()*10;
    b = rand()*10;

    for i=1:max_iterations
        a_new = a - learning_rate * dl_by_da(samples, a, b);
        b_new = b - learning_rate * dl_by_db(samples, a, b);

        if ( abs(a_new - a) < max_acc && abs(b_new - b) < max_acc )
            #print("Solution Converged");
            break;
        end

        a = a_new;
        b = b_new;

    end

    return a, b;
end
```

Out[4]: gradient_desent_beta (generic function with 1 method)

In [5]:
```
d = Beta(2,2);
samples = rand(d, 1000);
n = length(samples)
```

Out[5]: 1000

In [6]:
```
max_iterations = 10000000;
learning_rate = 0.00001;

a, b = gradient_desent_beta(samples, learning_rate, max_iterations)
```

Out[6]: (1.928375306495845, 1.987824660444101)

In [7]:
```
### The estimated values of the Beta distribution Parameters are ###
alpha = 1.928
Beta  = 1.987
```

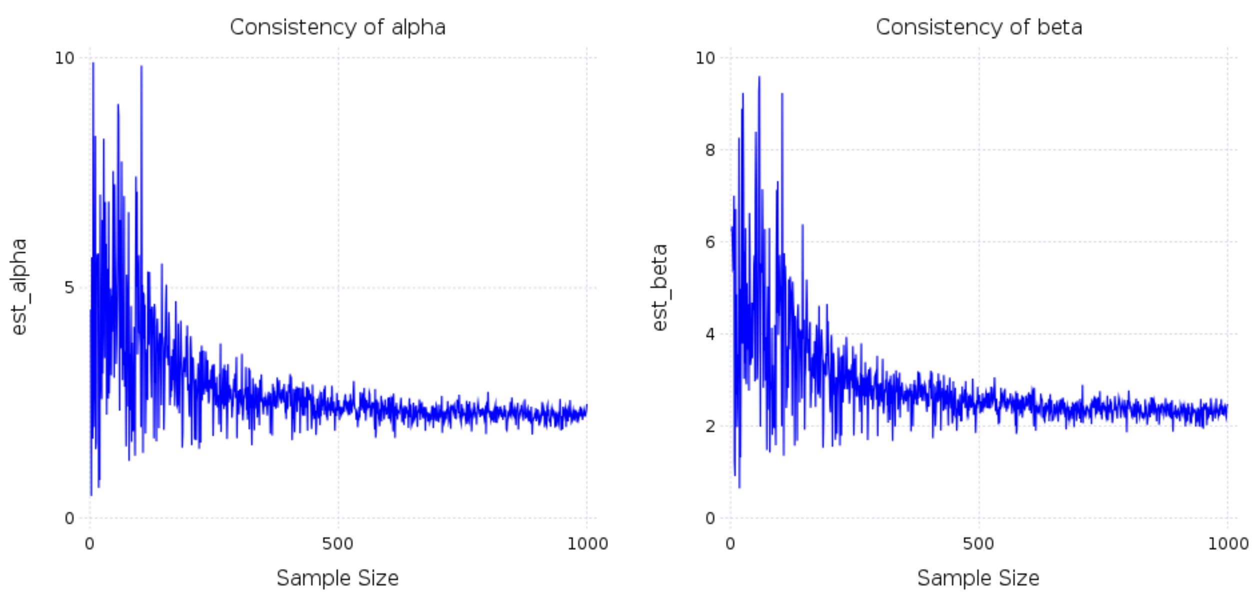
```
WARNING: imported binding for Beta overwritten in module Main
```

Out[7]: 1.987

In [15]:
```
d = Beta(a, b);
samples_size = collect(2:1000);

estimate_a = zeros(length(samples_size));
estimate_b = zeros(length(samples_size));

for i=1:length(samples_size)
    samples = rand(d,samples_size[i]);
    estimate_a[i], estimate_b[i] = gradient_desent_beta(samples, learning_rate, max_iterations)
end
white_panel = Theme(panel_fill=colorant"white", default_color=colorant"blue", major_label_font_size =12p
myplot1 = Gadfly.plot(x=samples_size,y=estimate_a,Geom.line,
              Guide.xlabel("Sample Size"), Guide.ylabel("est_alpha"),
              Guide.title("Consistency of alpha"),white_panel);
myplot2 = Gadfly.plot(x=samples_size,y=estimate_b,Geom.line,
              Guide.xlabel("Sample Size"), Guide.ylabel("est_beta"),
              Guide.title("Consistency of beta"),white_panel);
final_plot = hstack(myplot1,myplot2);
draw(PNG(10inch, 5inch), final_plot);
```




In [ ]:
```
### The parameters are converging to the original values hence it is consistent ###
```