Toan Ly

Lecture 8 Exercise

1)

a) $L(\alpha,\beta) = p(x_1, x_2, \ldots, x_n|\alpha,\beta) = \prod_{i=1}^{n} \frac{\Gamma(\alpha+\beta)}{\Gamma(\alpha)\Gamma(\beta)} x_i^{\alpha-1}(1-x_i)^{\beta-1}$

$$= \frac{\Gamma(\alpha+\beta)^n}{\Gamma(\alpha)^n\Gamma(\beta)^n} \prod_{i=1}^{n} x_i^{\alpha-1}\,(1-x_i)^{\beta-1}$$

b) Log-likelihood function:

$$\ell(\alpha,\beta) = logL(\alpha,\beta)$$

$$= nlog\big(\Gamma(\alpha+\beta)\big) - nlog\big(\Gamma(\alpha)\big) - nlog\big(\Gamma(\beta)\big)$$

$$+ (\alpha-1)\sum_{i=1}^{n} logx_i + (\beta-1)\sum_{i=1}^{n} \log(1-x_i)$$

Negative log-likelihood function:

$$L = -\ell(\alpha,\beta) = n\Big(\log\big(\Gamma(\alpha)\big) + \log\big(\Gamma(\beta)\big) - \log\big(\Gamma(\alpha+\beta)\big)\Big)$$

$$+ (1-\alpha)\textstyle\sum_{i=1}^{n} logx_i + (1-\beta)\sum_{i=1}^{n} \log(1-x_i)$$

c)

$$\frac{\partial L}{\partial \alpha} = n\frac{\partial}{\partial \alpha}\log\big(\Gamma(\alpha)\big) - n\frac{\partial}{\partial \alpha}\log\big(\Gamma(\alpha+\beta)\big) - \sum_{i=1}^{n} logx_i$$

$$\frac{\partial L}{\partial \beta} = n\frac{\partial}{\partial \beta}\log\big(\Gamma(\beta)\big) - n\frac{\partial}{\partial \beta}\log\big(\Gamma(\alpha+\beta)\big) - \sum_{i=1}^{n} log(1-x_i)$$

d) Gradient descent algorithm:

- Initialize α_0, β_0
- Assign learning rate γ, max iteration maxIter
- For t = 1 : maxIter
    - $\alpha_t = \alpha_{t-1} - \gamma \frac{\partial L}{\partial \alpha}$
      $$= \alpha_{t-1} - \gamma \left(n \frac{\partial}{\partial \alpha} \log\big(\Gamma(\alpha)\big) - n \frac{\partial}{\partial \alpha} \log\big(\Gamma(\alpha+\beta)\big) - \sum_{i=1}^{n} log x_i \right)$$
    - $\beta_t = \beta_{t-1} - \gamma \ \frac{\partial L}{\partial \beta}$
      $$= \beta_{t-1} - \gamma \left(n \frac{\partial}{\partial \beta} \log\big(\Gamma(\beta)\big) - n \frac{\partial}{\partial \beta} \log\big(\Gamma(\alpha+\beta)\big) - \sum_{i=1}^{n} log(1 - x_i) \right)$$
    - If ($\alpha_t < 0$), reinitialize α_t since α_t has to be greater than 0
    - Same for β_t
    - If $|\alpha_t - \alpha_{t-1}| < \epsilon \ and \ |\beta_t - \beta_{t-1}| < \epsilon$, terminate for loop
- End for
- Return α, β

e)

```julia
d = Beta(2, 2)
sample = rand(d, 1000)
```

```julia
function dL_da(sample, a, b)
    n = length(sample);
    return n * (digamma(a) - digamma(a + b)) - sum(log.(sample));
end

function dL_db(sample, a, b)
    n = length(sample);
    return n * (digamma(b) - digamma(a + b)) - sum(log.(1 .- sample));
end
```

```julia
function grad_descent_beta(sample)
    n = length(sample);
    max_iter = 1000;
    epsilon = 1e-3;
    lr = 1e-3;

    # Initialize parameters
    alpha = rand() * 10;
    beta = rand() * 10;

    for i = 1:max_iter
        alpha_new = alpha - lr * dL_da(sample, alpha, beta);
        beta_new = beta - lr * dL_db(sample, alpha, beta);
        # Ensure parameters remain positive
        alpha_new = max(alpha_new, 1e-3);
        beta_new = max(beta_new, 1e-3);

        if abs(alpha_new - alpha) < epsilon && abs(beta_new - beta) < epsilon
            alpha = alpha_new;
            beta = beta_new;
            # println("Converged in iteration $i");
            # println("Estimated parameters: alpha = $alpha, beta = $beta");
            break;
        end
        alpha = alpha_new;
        beta = beta_new;
    end
    return alpha, beta;
end
```

```julia
grad_descent_beta(sample)
```

✓ 0.0s

```
(2.111670869420904, 2.1709783392577395)
```

f)

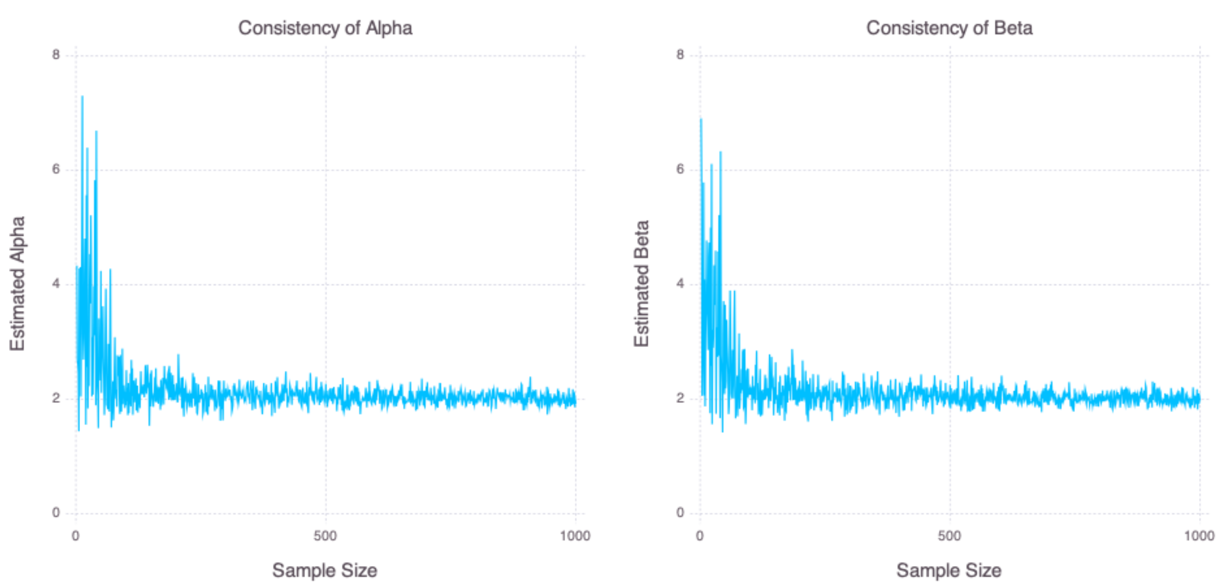
```
sample_sizes = collect(2:1000)
est_alpha = zeros(length(sample_sizes))
est_beta = zeros(length(sample_sizes))

for i = 1:length(sample_sizes)
    sample = rand(d, sample_sizes[i])
    a, b = grad_descent_beta(sample)
    est_alpha[i] = a
    est_beta[i] = b
end

myplot1 = plot(x=sample_sizes, y=est_alpha, Geom.line,
    Guide.xlabel("Sample Size"),
    Guide.ylabel("Estimated Alpha"),
    Guide.title("Consistency of Alpha"),
    Theme(background_color="white")
)

myplot2 = plot(x=sample_sizes, y=est_beta, Geom.line,
    Guide.xlabel("Sample Size"),
    Guide.ylabel("Estimated Beta"),
    Guide.title("Consistency of Beta"),
    Theme(background_color="white")
)

final_plot = hstack(myplot1, myplot2)
draw(PNG("estimation.png", 10inch, 5inch), final_plot)
```



Observation:

For small sample size < 100, $\hat{\alpha}\ and\ \hat{\beta}$ seems to be noisy and deviates from true value 2.0. Increasing the sample size helps them converge to the true value (2.0, 2.0), with small deviation around the true value. Therefore, Beta Distribution is consistent when increasing the sample size.