Problem Set II: Decision Theory

Jaeho Cho

Results

```
(d) Running the experiment for each case...


Running experiments for each case with N = 1...
Case I: b1 = 2, r1 = 8; b2 = 2, r2 = 8
Results for Case I:
Theoretical P_error (ML): 0.2909
Empirical P_error (ML): 0.0000
Theoretical P_error (MAP): 0.2000
Empirical P_error (MAP): 0.0000


Case II: b1 = 6, r1 = 4; b2 = 2, r2 = 8
Results for Case II:
Theoretical P_error (ML): 0.5091
Empirical P_error (ML): 0.0000
Theoretical P_error (MAP): 0.4000
Empirical P_error (MAP): 0.0000


Case III: b1 = 2, r1 = 8; b2 = 6, r2 = 4
Results for Case III:
Theoretical P_error (ML): 0.5091
Empirical P_error (ML): 0.0000
Theoretical P_error (MAP): 0.2000
Empirical P_error (MAP): 0.0000


Case IV: b1 = 6, r1 = 4; b2 = 6, r2 = 4
Results for Case IV:
Theoretical P_error (ML): 0.4364
Empirical P_error (ML): 1.0000
Theoretical P_error (MAP): 0.4000
Empirical P_error (MAP): 0.0000


(f) Running the experiment N = 1e5 times for each case...


Running experiments for each case with N = 100000...
Case I: b1 = 2, r1 = 8; b2 = 2, r2 = 8
Results for Case I:
Theoretical P_error (ML): 0.2909
Empirical P_error (ML): 0.2925
```

```
Theoretical P_error (MAP): 0.2000
Empirical P_error (MAP): 0.2006


Case II: b1 = 6, r1 = 4; b2 = 2, r2 = 8
Results for Case II:
Theoretical P_error (ML): 0.5091
Empirical P_error (ML): 0.5087
Theoretical P_error (MAP): 0.4000
Empirical P_error (MAP): 0.3993


Case III: b1 = 2, r1 = 8; b2 = 6, r2 = 4
Results for Case III:
Theoretical P_error (ML): 0.5091
Empirical P_error (ML): 0.5085
Theoretical P_error (MAP): 0.2000
Empirical P_error (MAP): 0.1986


Case IV: b1 = 6, r1 = 4; b2 = 6, r2 = 4
Results for Case IV:
Theoretical P_error (ML): 0.4364
Empirical P_error (ML): 0.4364
Theoretical P_error (MAP): 0.4000
Empirical P_error (MAP): 0.4011
```

Matlab Code

```
%%%% Problem Set II: Decision Theory
% Jaeho Cho, Sep 23

clc; clear; close all;

%% (d) Run the experiment for each case
fprintf('(d) Running the experiment for each case...\n\n');
run_all_cases(1);

%% (e) Decision Rule: ML vs. MAP
% One of these decision rules (either ML or MAP) is the same in all cases.
% Which one? Why does this happen given how we have set up the experiment?
%
% *Answer:* The ML decision rule is the same in all cases because it depends
% only on the likelihood functions. Since the probability of observing
% RedII given RedI is always greater than that of observing RedII given BlueI,
% ML always decides RedI when observing RedII, and vice versa.

%% (f) Run the experiment N = 1e5 times for each case
% *Comparison:* The theoretical and empirical probabilities of error are
% relatively close to each other.
fprintf('(f) Running the experiment N = 1e5 times for each case...\n\n');
run_all_cases(1e5);

%% Function to run all experiment cases
function run_all_cases(N)
    fprintf('Running experiments for each case with N = %.0f...\n', N);

    run_case(2, 8, 2, 8, N, 'Case I');
    run_case(6, 4, 2, 8, N, 'Case II');
    run_case(2, 8, 6, 4, N, 'Case III');
    run_case(6, 4, 6, 4, N, 'Case IV');
end

%% Function to run one experiment case
function run_case(b1, r1, b2, r2, N, case_name)
    fprintf('%s: b1 = %d, r1 = %d; b2 = %d, r2 = %d\n', case_name, b1, r1, b2, r2);

    % Compute prior, likelihood, and posterior probabilities
    [prior, likelihood, posterior] = compute_probabilities(r1, b1, r2, b2);

    % Compute ML and MAP decision rules
    ML_decision = ML_decision_rule(likelihood);
    MAP_decision = MAP_decision_rule(posterior);

    % Compute theoretical probability of error
    P_error_ML = compute_probability_of_error(ML_decision, prior, likelihood, posterior);
    P_error_MAP = compute_probability_of_error(MAP_decision, prior, likelihood, posterior);

    % Simulate the experiment N times
    empirical_error_ML = simulate_experiment(N, r1, b1, r2, b2, ML_decision);
    empirical_error_MAP = simulate_experiment(N, r1, b1, r2, b2, MAP_decision);

    % Display results for the current case
    fprintf('Results for %s:\n', case_name);
    fprintf('Theoretical P_error (ML): %.4f\n', P_error_ML);
    fprintf('Empirical P_error (ML): %.4f\n', empirical_error_ML);
    fprintf('Theoretical P_error (MAP): %.4f\n', P_error_MAP);
    fprintf('Empirical P_error (MAP): %.4f\n\n', empirical_error_MAP);
end

%% (a) Function to compute prior, likelihood, and posterior probabilities
function [prior, likelihood, posterior] = compute_probabilities(r1, b1, r2, b2)
    % Prior probabilities
    total_I = r1 + b1;
    prior.RedI = r1 / total_I;
    prior.BlueI = b1 / total_I;
    % Likelihood functions
    total_II = r2 + b2 + 1;

    likelihood.RedII_given_RedI = (r2 + 1) / total_II;
    likelihood.BlueII_given_RedI = b2 / total_II;

    likelihood.RedII_given_BlueI = r2 / total_II;
    likelihood.BlueII_given_BlueI = (b2 + 1) / total_II;

    % Posterior probabilities
    P_RedII = likelihood.RedII_given_RedI * prior.RedI + likelihood.RedII_given_BlueI * prior.BlueI;
    P_BlueII = 1 - P_RedII;

    posterior.RedI_given_RedII = (likelihood.RedII_given_RedI * prior.RedI) / P_RedII;
    posterior.BlueI_given_RedII = (likelihood.RedII_given_BlueI * prior.BlueI) / P_RedII;

    posterior.RedI_given_BlueII = (likelihood.BlueII_given_RedI * prior.RedI) / P_BlueII;
    posterior.BlueI_given_BlueII = (likelihood.BlueII_given_BlueI * prior.BlueI) / P_BlueII;
end

%% (b.1) ML decision rule function
function ML_decision = ML_decision_rule(likelihood)
    if likelihood.RedII_given_RedI > likelihood.RedII_given_BlueI
        ML_decision(1) = 1; % Decide RedI when observing RedII (always for ML)
    else
        ML_decision(1) = 2; % Decide BlueI when observing RedII
    end

    if likelihood.BlueII_given_BlueI > likelihood.BlueII_given_RedI
        ML_decision(2) = 2; % Decide BlueI when observing BlueII (always for ML)
    else
        ML_decision(2) = 1; % Decide RedI when observing BlueII
    end
end

%% (b.2) MAP decision rule function
function MAP_decision = MAP_decision_rule(posterior)
    if posterior.RedI_given_RedII > posterior.BlueI_given_RedII
        MAP_decision(1) = 1; % Decide RedI when observing RedII
    else
        MAP_decision(1) = 2; % Decide BlueI when observing RedII
    end

    if posterior.BlueI_given_BlueII > posterior.RedI_given_BlueII
        MAP_decision(2) = 2; % Decide BlueI when observing BlueII
    else
```

```
        MAP_decision(2) = 1; % Decide RedI when observing BlueII
    end
end

%% (c) Function to compute the probability of error
function P_error = compute_probability_of_error(decision_rule, prior, likelihood, posterior)
    P_RedII = likelihood.RedII_given_RedI * prior.RedI + likelihood.RedII_given_BlueI * prior.BlueI;
    P_BlueII = 1 - P_RedII;

    if decision_rule(1) == 1    % Decide RedI when observing RedII
        P_error_RedII = P_RedII * posterior.BlueI_given_RedII;
    else                        % Decide BlueI when observing RedII
        P_error_RedII = P_RedII * posterior.RedI_given_RedII;
    end

    if decision_rule(2) == 1    % Decide RedI when observing BlueII
        P_error_BlueII = P_BlueII * posterior.BlueI_given_BlueII;
    else                        % Decide BlueI when observing BlueII
        P_error_BlueII = P_BlueII * posterior.RedI_given_BlueII;
    end

    P_error = P_error_RedII + P_error_BlueII;
end

%% Function to simulate the experiment N times and compute empirical error
function empirical_error = simulate_experiment(N, r1, b1, r2, b2, decision_rule)
    error_count = 0;
    for i = 1:N
        [ball_I, ball_II] = experiment_instance(r1, b1, r2, b2);
        decision = decision_rule(ball_II);
        if decision ~= ball_I
            error_count = error_count + 1;
        end
    end
    empirical_error = error_count / N;
end

%% Function to simulate an instance of the experiment
function [ball_I, ball_II] = experiment_instance(r1, b1, r2, b2)
    ball_I = choose_ball(r1, b1);

    if ball_I == 1
        r2_updated = r2 + 1;
        b2_updated = b2;
    else
        r2_updated = r2;
        b2_updated = b2 + 1;
    end

    ball_II = choose_ball(r2_updated, b2_updated);
end
```

```
%% Function to pick a ball from an urn with r red balls and b blue balls
function chosen_color = choose_ball(r, b)
    ball = randi([1 r + b]);

    if ball <= r
        chosen_color = 1; % Red ball
    else
        chosen_color = 2; % Blue ball
    end
end
```