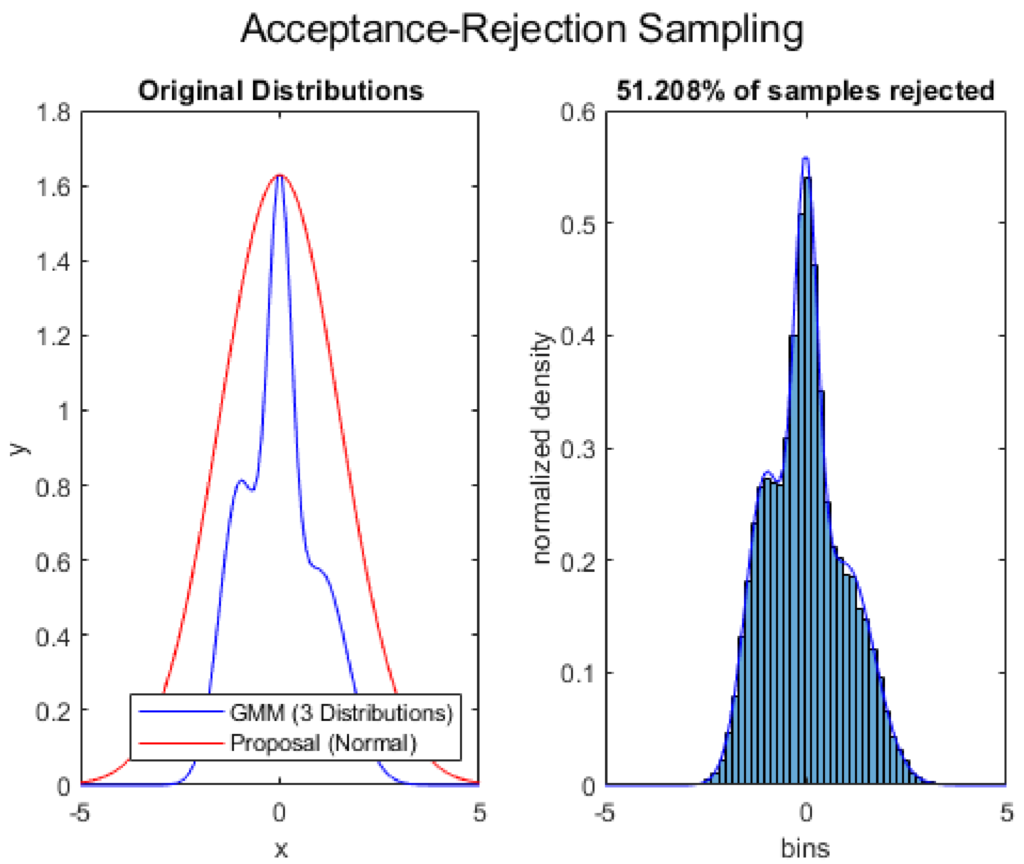
Contents


```
% ECE414 - Bayesian Machine Learning
% Authors   : Junbum Kim, Andy Jeong
% Project 6 : Markov Chain Monte Carlo Sampling Methods
% Date      : December 11, 2019
% Reference : Pattern Recognition and Machine Learning by C. M. Bishop (2006)
close all; clear all; clc;     % clear workspace variables
rng(42);                       % for reproducibility
```

Acceptance-Rejection Sampling

```
% parameters for 3 Gaussian distributions
mu = [-1, 0, 1];             % means
sigma = [0.5, 0.3, 0.7];     % standard deviation
N = 100;

% anonymous functions
% Gaussian Mixture Model (3)  % add pdf's
p_fcn = @(x) normpdf(x, mu(1), sigma(1)) ...
             + normpdf(x, mu(2), sigma(2)) ...
             + normpdf(x, mu(3), sigma(3));
% Proposal Distribution (normal)
q_fcn = @(x) normpdf(x, mean(mu), sum(sigma));

% draw pdf's from GMM and Proposal and compute the constant k
% constant k is for accepting in proportion to the highest value from GMM
x = linspace(-5,5,N);
p_samples = normpdf(x, mean(mu), sum(sigma)); % proposed samples
data = p_fcn(x);                              % samples from GMM
k = max(data./p_samples); % ~6.1

% plot GMM and proposal distributions
figure(1); subplot(1,2,1);
plot(x, data, 'b'); hold on;
plot(x, k*p_samples,'r'); hold off;
legend('GMM (3 Distributions)', 'Proposal (Normal)','Location','Southeast');
xlabel('x'); ylabel('y');
title('Original Distributions');

% perform sampling for the specified iteration times
samples = [];
iterations = 100000;
accepted = 0;
for i = 1:iterations
    % draw random variables from proposal distribution
    z = normrnd(mean(mu), sum(sigma));
    % draw from uniform distribution ~U(0, k*q(z))
    % 1e4 is to get a value between 0 and 1 with 4 decimal place precision
    u = randi(round(k*q_fcn(z)*1e4))/1e4;
    if u <= p_fcn(z)
       samples(end+1) = z;
       accepted = accepted + 1;
    end
```

```
end

% plot histograms of the samples
num_bins = 40;
figure(1); subplot(1,2,2);
histogram(samples, num_bins, 'Normalization', 'pdf'); hold on;
xlabel('bins'); ylabel('normalized density');
C = 1/k*2.1; % constant multiplied to pdf to trace the envelope of histograms
plot(x, data .* C, 'b');
title([num2str((1-accepted/iterations)*100) '% of samples rejected']);

suptitle('Acceptance-Rejection Sampling');
```

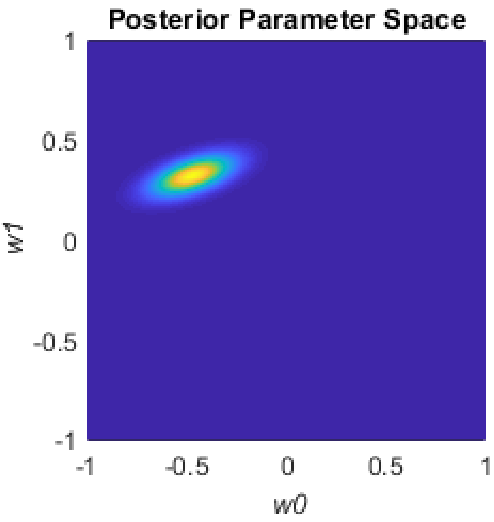


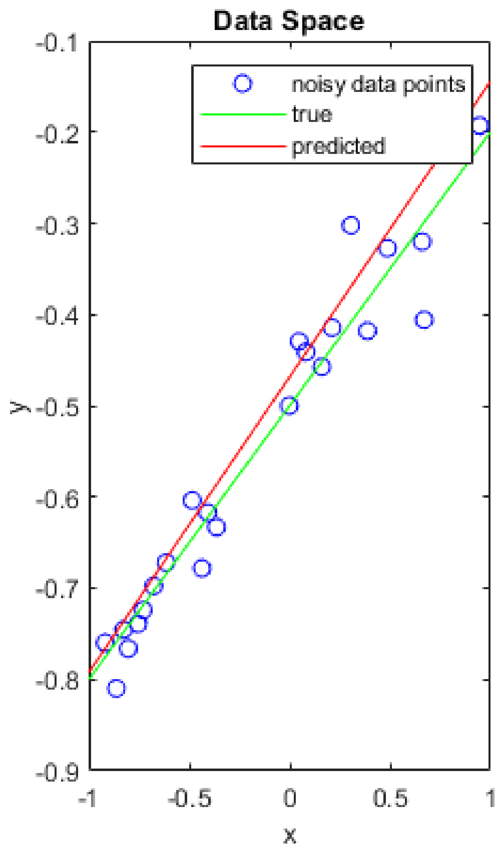
MCMC Estimation

```
% number of observations, sample iterations and burn-ins
N = 25;
steps = 1e5;
burnin = 100;

% weights for true distribution
w_true = [-0.5, 0.3];

% hyperparameter
sd = 5e-2;           % standard deviation
beta = (1 / sd)^2; % precision

% generate noisy sample data points to a line
x_data = rand([1,N]) * 2 - 1;
y_data = w_true * [ones([1,N]) ; x_data] + normrnd(0, sd, [1,N]);

% Parameters for proposal distribution
```

```
% **mean weights are sampled in each iteration
cov_prop = eye(2) * 1/100;

% Parameters for prior distribution
w_prior = [0 0];
cov_prior = eye(2) / 10;

% (from project 2) estimate mean vector and covariance for posterior
phi = [ones([1,N]); x_data]';
cov_post = inv(inv(cov_prior) + beta * phi' * phi);
w_post = (cov_post*(inv(cov_prior) * w_prior' + beta * phi' * y_data'))';

% initialize weights
w_init = [0, 0];
w_curr = w_init;

% run through 'steps' number of iterations
w_accepted = [];
for i = 1:steps
    % suggest a new position based on current mean value
    z = mvnrnd(w_curr, cov_prop); % new proposal sample

    % generate pdf of posterior distributions
    p_star = mvnpdf(z, w_post, cov_post);       % proposal posterior
    p_tau = mvnpdf(w_curr, w_post, cov_post);   % current posterior

    % accepting criterion
    A = min([1, p_star/p_tau]);

    % decide whether to accept or reject after burn-in
    % by comparing a random sample from ~U(0,1) to accepting criterion
    if rand < A && i > burnin
        w_accepted(:,end+1) = z;
        w_curr = z;
    end
end

% prepare a grid space
sampling_rate = 100;
[w0, w1] = meshgrid(linspace(-1,1,sampling_rate));
w = [w0(:), w1(:)];

% take mean and covariance of all accepted samples for the weights
w_mean = mean(w_accepted, 2);
w_cov = cov(w_accepted');
prob = mvnpdf(w, w_mean', w_cov); % generate pdf from the weights on grid
prob_grid = reshape(prob, [sampling_rate, sampling_rate]);

% plot the posterior distribution estimation
figure(2); subplot(1,2,1);
pcolor(w0, w1, prob_grid); shading interp; pbaspect([1 1 1]);
xlabel('{\it w0}'); ylabel('{\it w1}')
title('Posterior Parameter Space')

% plot true and predicted in data space
subplot(1,2,2);
x = linspace(-1,1,sampling_rate);
y_truth = w_true * [ones([1,sampling_rate]); x];
y_pred = w_mean' * [ones([1,sampling_rate]); x];
plot(x_data, y_data,'bo', x, y_truth, 'g', x, y_pred, 'r');
xlabel('x'); ylabel('y'); title('Data Space')
```

```
legend('noisy data points', 'true', 'predicted');
```





Published with MATLAB® R2018a