Module 5 Assignment on Resampling

Jingwen Zhong // Graduate Student

03/17

Module Assignment Questions

Q1) (*Random Variable Generation*)

You have learned how to generate exponential random variables using uniform distribution. Now, you will generate normal random variables using one of the methods described in this LINK.

Some useful information from the link:

- Normal distribution is also called as the Gaussian distribution.For $\mu = 0$ and $\sigma = 1$, we refer to this distribution as the standard normal distribution
- Normal random variables X 's with mean μ and variance 2 σ are generated by the relationship $X = \mu + \sigma Z$, where Z is the standard normal random variable.

```
set.seed(99)
# I use Method 9: Proposed Method 1. Generate U from the U(0,1) distribution.
unif_rv = runif(n = 1000, min = 0, max = 1)
# Return Z Z_from_me = -(log(1/unif_rv-1))/1.702 # log computes natural
# logarithms(ln=log_e) by default
Z_from_me = 0.46615 + 90.72192 * tanh(-31.35694 + 28.77154 * unif_rv) - 89.36967 *
    tanh(-2.57136 - 31.16364 * unif_rv) - 96.55499 * tanh(3.94963 - 1.66888 * unif_rv) +
    97.36346 * tanh(2.31229 + 1.84289 * unif_rv)
```

```
set.seed(99)
# rnorm: The Normal Distribution
Z = rnorm(1000, mean = 0, sd = 1)
```

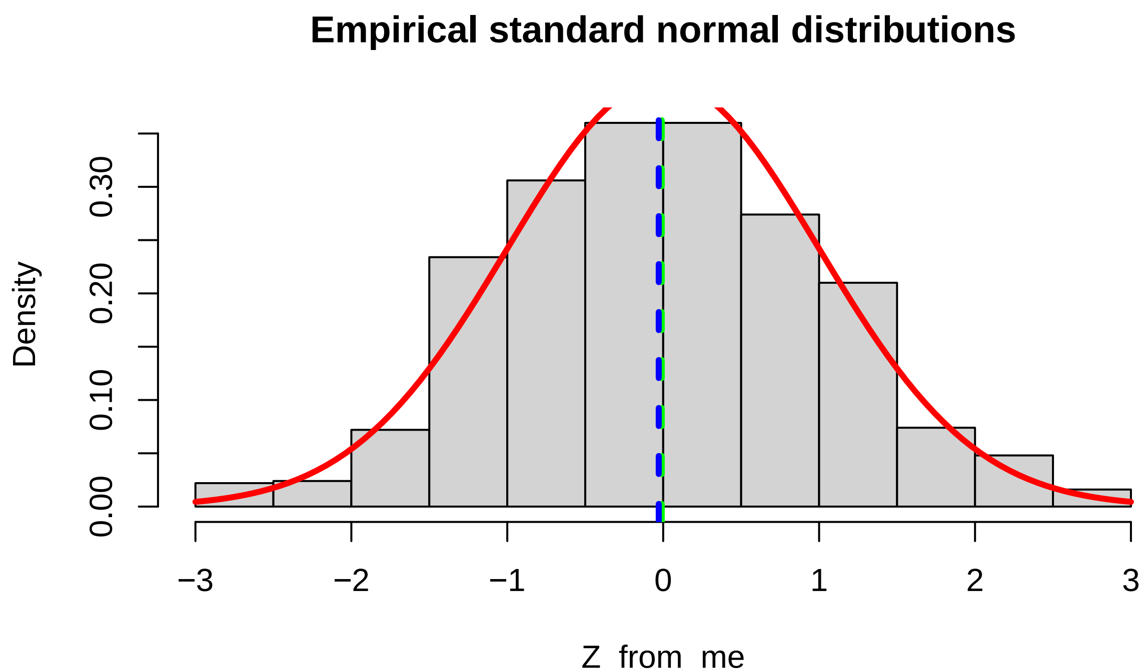
a. As we studied the exponential random variable in the lab session, , generate 1,000 random variables from the standard normal distribution using the method you chose in the link.

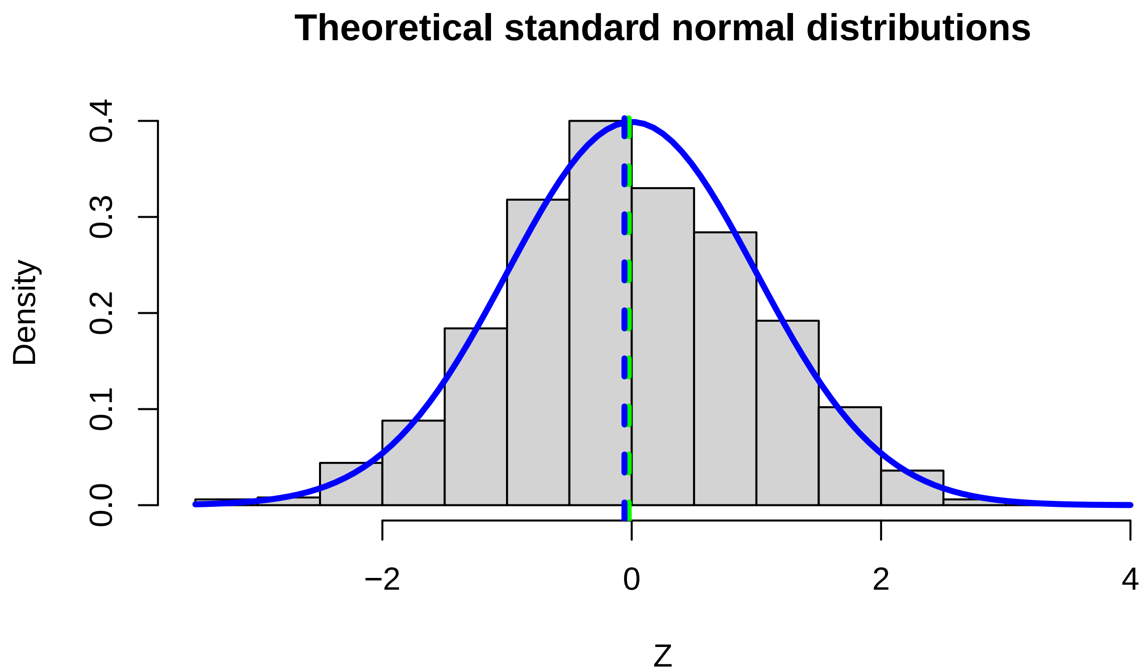
---

b. Check if this generation is verified with the theory:

1) check if the characteristics (so calculate mean, sd, skewness, kurtosis for each) of empirical and theoretical distributions are similar;

```
hist(Z_from_me, main = "Empirical standard normal distributions", prob = TRUE)
curve(dnorm(x, 0, 1), add = TRUE, col = "red", lwd = 3)
abline(v = c(mean(Z_from_me), median(Z_from_me)), col = c("green", "blue"), lty = c(2,
    2), lwd = c(3, 3))

hist(Z, main = "Theoretical standard normal distributions", prob = TRUE)
curve(dnorm(x, 0, 1), add = TRUE, col = "blue", lwd = 3)
abline(v = c(mean(Z), median(Z)), col = c("green", "blue"), lty = c(2, 2), lwd = c(3,
    3))
```





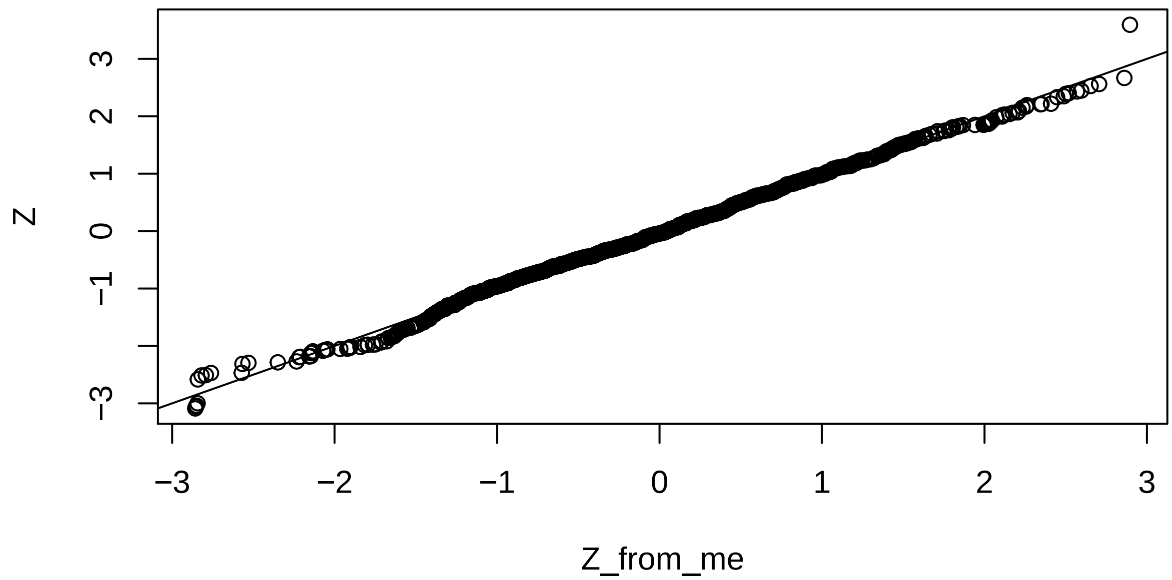

```
skewness1 = sum((Z_from_me - mean(Z_from_me))^3)/((length(Z_from_me) - 1) * sd(Z_from_me)^3)
kurtosis1 = sum((Z_from_me - mean(Z_from_me))^4)/((length(Z_from_me) - 1) * sd(Z_from_me)^4)

skewness2 = sum((Z - mean(Z))^3)/((length(Z) - 1) * sd(Z)^3)
kurtosis2 = sum((Z - mean(Z))^4)/((length(Z) - 1) * sd(Z)^4)

table = matrix(c(mean(Z_from_me), mean(Z), sd(Z_from_me), sd(Z), skewness1, skewness2,
    kurtosis1, kurtosis2), 2, 4)

colnames(table) = c("mean", "sd", "skewness", "kurtosis")
rownames(table) = c("Empirical distributions", "Theoretical distributions")

knitr::kable(table, caption = " Characteristics check")
```

Table 1: Characteristics check

	mean	sd	skewness	kurtosis
Empirical distributions	-0.0092624	1.024916	0.0445511	2.837735
Theoretical distributions	-0.0240978	1.022850	0.0040306	2.863798

2) plot a QQ and judge by eye

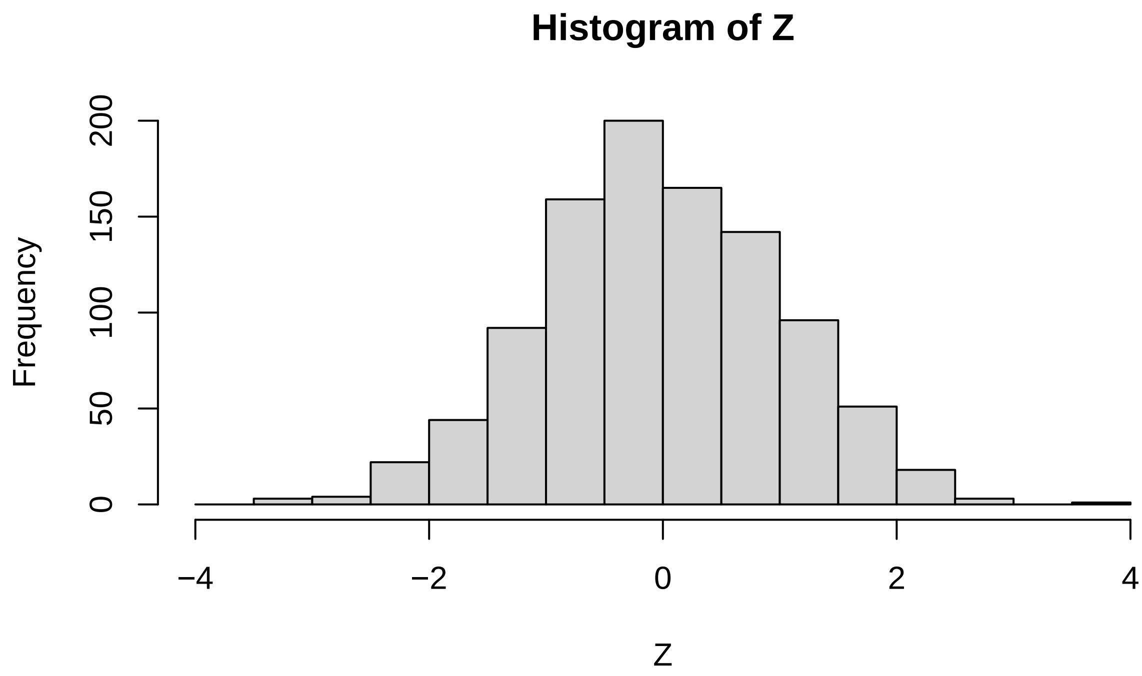
```
qqplot(Z_from_me, Z)
abline(0, 1)
```

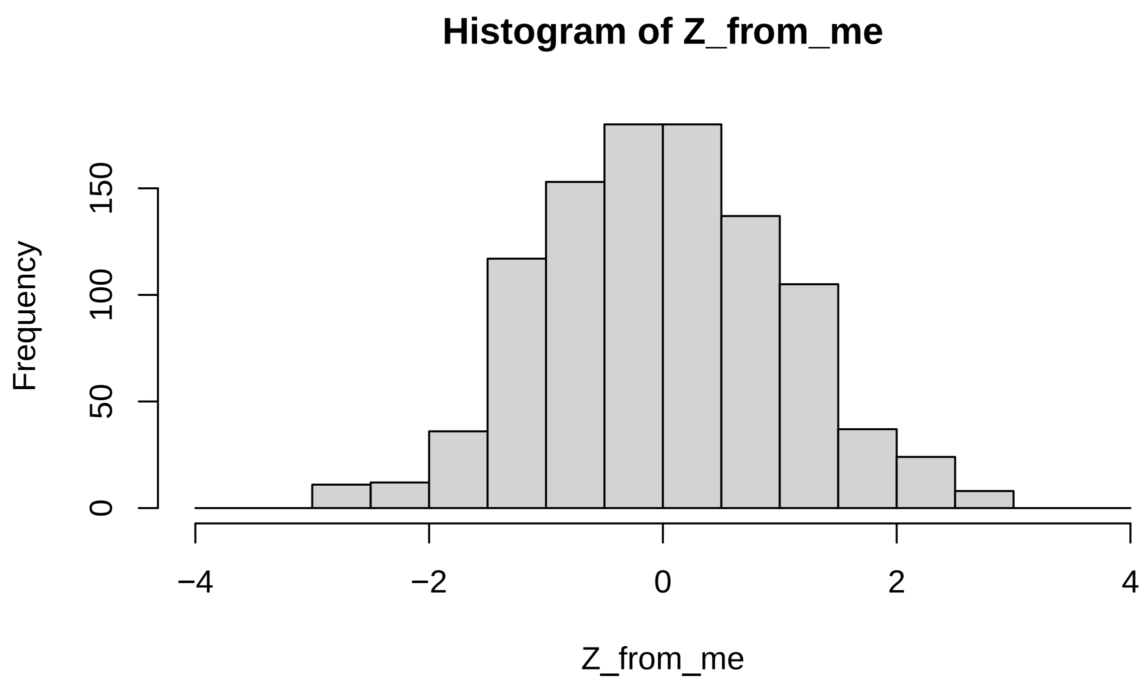


3) make a chi-squared test on empirical and theoretical buckets.

```
# use round(min(Z_from_me)) to round(max(Z_from_me)) for normal dist
bins = seq(-4, 4, by = 0.5)
# df is bins-1: ts=(sum(hist2-hist1)^2/(hist1)), p-value=1-P(X<ts) using chi-sq
# dist
hist1 = hist(Z, freq = TRUE, breaks = bins)$counts + 1
hist2 = hist(Z_from_me, freq = TRUE, breaks = bins)$counts + 1
chisq.test(hist2, p = hist1, rescale.p = TRUE)
```

```
##
##  Chi-squared test for given probabilities
##
## data:  hist2
## X-squared = 41.535, df = 15, p-value = 0.0002649
```





---

c. Write an overall comment on your experience.

I think my empirical distribution is very good. The mean, sd, skewness, kurtosis are all approximate to the theoretical standard normal distributions. Although the p-value of the Chi-sqaure test is too small so that we can reject $H_0 : Emperical = Theoretical$ and conclude that my empirical distribution is not standard normal distribution. In fact, I think using the seed(99), my empirical distribution performs better than the theoretical distributions.

Q2) (*Bootstrapping*)

Let $\hat{\sigma}_1^2$ and $\hat{\sigma}_2^2$ represent the variance estimates of two independent random samples of size 10 and 20, respectively, taken from any normal distributions with variances $\sigma_1^2 = 12$ and $\sigma_2^2 = 8$, respectively. Let a new parameter be defined as $v = \frac{\sigma_1^2}{\sigma_2^2}$.

Based on randomly generated samples with any choice on means, use bootstrap method (B=1000) to answer each part below:

```
library(boot)

# define parameters
set.seed(99)
x1 = rnorm(1000, mean = 0, sd = sqrt(12))
x2 = rnorm(1000, mean = 0, sd = sqrt(8))
df = cbind(x1, x2)
```

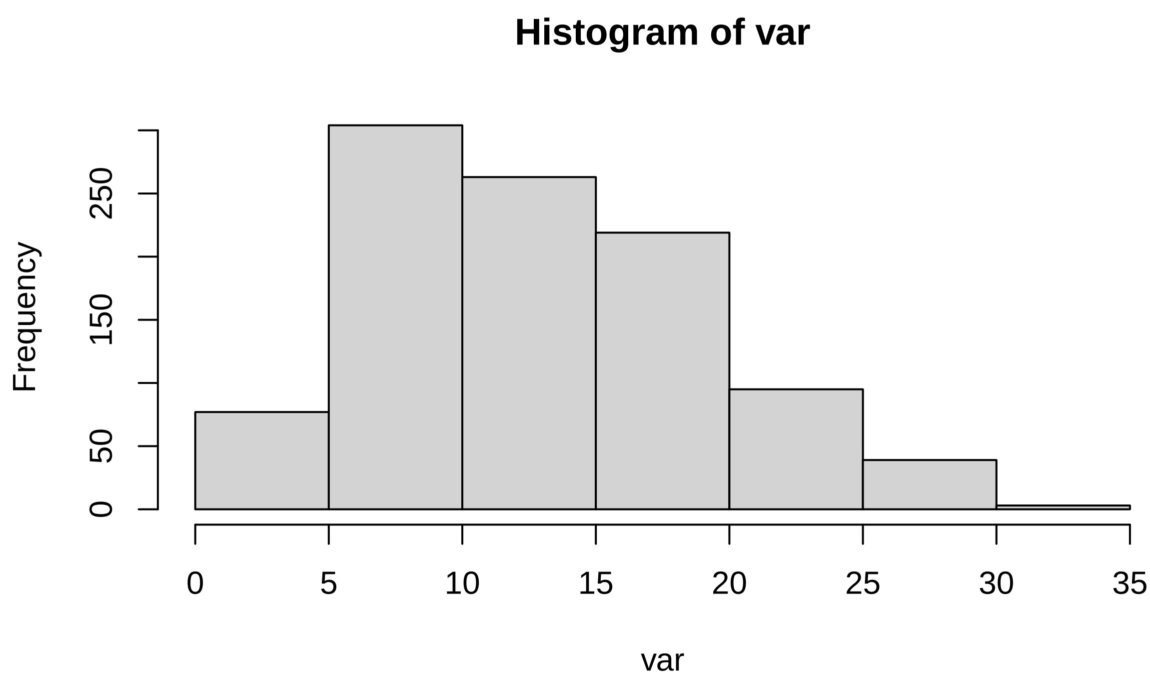
```
var1 <- function(data, i) {
    var <- sd(data[i])^2  # variance for first normal distribution
    return(var)
}

set.seed(99)
data1 = sample(df[, 1], 10)
results <- boot(data = data1, statistic = var1, R = 1000)
results
```

a. Estimate σ_1^2 along with the standard error on the estimate. Evaluate if this misses the true value.

```
##
## ORDINARY NONPARAMETRIC BOOTSTRAP
##
##
## Call:
## boot(data = data1, statistic = var1, R = 1000)
##
##
## Bootstrap Statistics :
##     original    bias    std. error
## t1* 14.24019 -1.319274    6.135351
```

```
var = results$t
hist(var)

cat("\nit is close to the true value", 12)
```

```
##
## it is close to the true value 12
```



```r
v = 12/8

var_ratio <- function(data, indices, n1 = 10, n2 = 20) {
    set.seed(99)
    sample1 = df[sample(size = n1, x = indices, replace = TRUE), 1]
    sample2 = df[sample(size = n2, x = indices, replace = TRUE), 2]
    ratio = (sd(sample1)/sd(sample2))^2
    return(ratio)
}

results2 <- boot(df, statistic = var_ratio, R = 1000)
results2
```

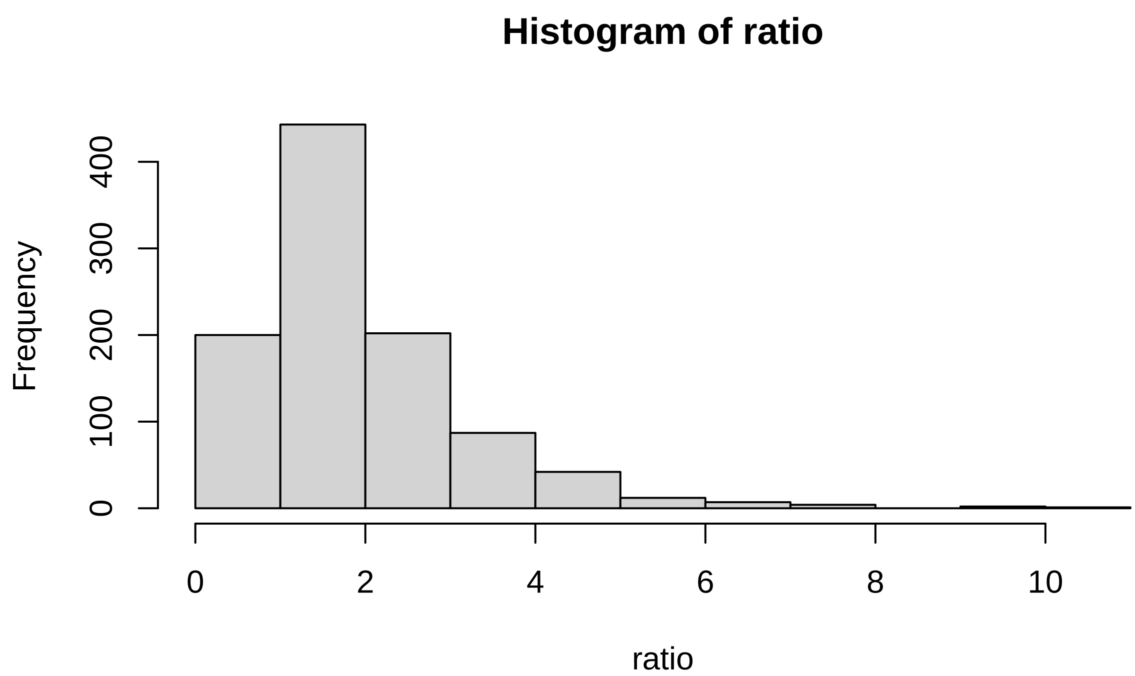
b. Estimate the parameter v along with the standard error on the estimate. Evaluate if this misses the true value.

```
##
## ORDINARY NONPARAMETRIC BOOTSTRAP
##
##
## Call:
## boot(data = df, statistic = var_ratio, R = 1000)
##
```

```
##
## Bootstrap Statistics :
##     original    bias    std. error
## t1* 1.528839 0.3954219    1.259916
```

```
ratio = results2$t[1:1000, ]
hist(ratio)

cat("\nit is close to the true value", v)
```

```
##
## it is close to the true value 1.5
```



```
boot.ci(results2, type = "bca")
```

c. Obtain the percentile-based 95% confidence interval on the v estimate.

```
## BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
## Based on 1000 bootstrap replicates
##
## CALL :
## boot.ci(boot.out = results2, type = "bca")
```

```
##
## Intervals :
## Level       BCa
## 95%   ( 0.404,  4.663 )
## Calculations and Intervals on Original Scale
```

---

d. (BONUS) Search for what theory says about the CI on the σ^2 and v estimates. Verify if the result in part c convinces the theory.

---

e. (BONUS) Can bootstrapping method solve the probability problem, $P(\frac{\hat{\sigma}_1^2}{\hat{\sigma}_2^2} > 2)$? Calculate and show. Does it give similar result from the theory?

Q3) (*Advanced Sampling*)

Watch `the playlist` on Hidden Markov, MC, MCMC, Gibbs, Metropolis-Hastings methods (or, use this link https://youtube.com/playlist?list=PLTpORyMWYJsa-bLjJqzsJKXQCrMAiYu7n)

a. Write one paragraph summary for each MCMC, Gibbs and Metropolis-Hastings method by highlighting how the strategy works (watch six videos, write three paragraphs). sample from P(x), (P(x)=f(x)/NC, but we don't know P(x), we only know f(x), NC=normalized constant)

- MCMC: Design a special Markov chain: Learn from the previous samples, and pick the next sample based on that, so samples are no longer independent. Monte carlo: Simulating samples/draws from our target distribution P(x), by simulating this markov chain, and eventually this markov chain is going to simulate draws from P(x) (throw away "burn-in" part).
- Gibbs: Use when we sample from a multivariate distribution. Sampling from conditional distributions(not joint distributions), such like P(x|y). For x, y example: Start by initial x, y, then change new x, keep y fixed, then sample new y keep new x fixed, and so on so forth.
- Metropolis-Hastings: step one(candidates): Center a (normal) distribution at the previous sample x_t and with some fixed variance, then sample the next candidate x_{t+1} from that normal distribution. step two: accept the candidate with some probabilities ($\frac{A(a->b)}{A(b->a)} = \frac{f(b)}{f(a)}\frac{g(a|b)}{g(b|a)} = r_f r_g$, acceptance probablity: $A(a->b) = MIN(1, r_f r_g)$).

---

b. (BONUS) Give an example from `Deep Learning` if any of them is employed.
HMM is employed in Deep Learning, for example: Siri

Write comments, questions: ...

Disclose the resources or persons if you get any help: ...

How long did the assignment solutions take?: 10 hrs

References: ...