

Monte Carlo, Markov chain methods and applications

- Group 17
 - G.Sreeja -S20170010047
 - T.Ooha-S20170010167
 - M.Manasa-S20170020218



Contents

- ❖ Monte Carlo methods
 - Direct Sampling
 - Application - Todo List
 - Importance sampling
 - Application - sampling from a complex function
 - Rejection sampling
 - Application - for reduction in the variance of an integral approximation.



Contd...

- ❖ Markov chain
 - Application - Predicting Next Few words for a given text
- ❖ Markov Chain Monte Carlo
 - Metropolis Hastings
 - Application - Decrypting Text
 - Simulated Annealing
 - Application - Knapsack Problem
 - Gibbs Sampling
 - Application - Spam filter



Monte Carlo Methods

- Monte Carlo Methods are a broad class of algorithms that rely on repeated random sampling to obtain numerical results.
- We use Monte Carlo methods to sample a probability distribution to approximate a quantity (mean or variance of distribution)



Types of Monte Carlo

- Monte Carlo methods are defined in terms of the way that samples are drawn or the constraints imposed on the sampling.
- There are types of Monte Carlo as follows :
 - Direct Sampling
 - Importance Sampling
 - Rejection Sampling

Direct sampling



Direct Sampling

- Direct sampling is a technique where samples are generated **directly** from $p(x)$ where $p(x)$ is probability density function .
- Also the **law of large numbers** states that if the samples $x(i)$ are i.i.d., then the average converges almost surely to the expected value .
- So from this we can observe that if we take more samples we get more accurate results

Contd...

- Let $p(x)$ be probability density function. Then expectation is

$$\mathbb{E}_{x \sim p}[f(x)] = \sum_x f(x)p(x).$$

- Monte Carlo technique approximates a target expectation with

$$\mathbb{E}_{x \sim p}[f(x)] = \frac{1}{T} \sum_{t=1}^T f(x^t),$$

where x^1, \dots, x^t are samples drawn from p . Also,

$$\begin{aligned}\mathbb{E}_{x^1, \dots, x^T \stackrel{\text{i.i.d.}}{\sim} p}[I_T] &= \mathbb{E}_{x \sim p}[f(x)] \\ \text{Var}_{x^1, \dots, x^T \stackrel{\text{i.i.d.}}{\sim} p}[I_T] &= \frac{1}{T} \text{Var}_{x \sim p}[f(x)]\end{aligned}$$



Application - Optimising Todo list

- Todo lists are more important in our daily situations. If you don't have enough time, which ones should you tackle first and what is the strategy to use to optimise your time?
- Here we are finding tasks completed in time, important tasks completed, tasks completed using monte carlo simulation.



Methodology

- Number of simulations=1000,Number of tasks=50
- Labels :Do as they come,Due tasks first,Due tasks last,Important tasks first,Easier tasks first,Easier important tasks first,Easier due tasks first
- For each simulation :
 - Create a task
 - Creating tasks : Each task has 4 attributes.
 - Weight : random number between 1 to 100 .
 - Due date: integer indicating the number of days away from the start of the simulation date.
 - Duration: random integer between 1 and the due date, using Randint.
 - done : set to 0 when the task is created.



Contd...

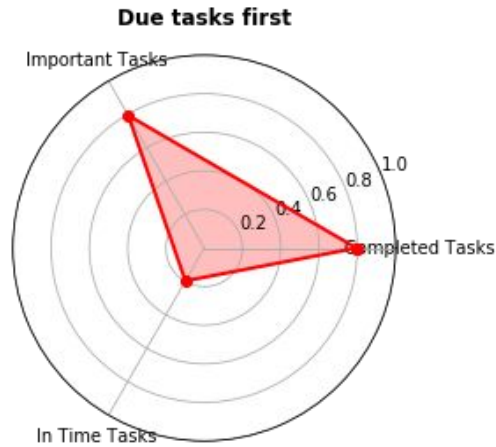
- **Assumptions:**
 - Deadline is 50% of sum of durations of all tasks
 - More weight indicates more important the task is.
 - 75th percentile of tasks are considered as important tasks after sorting tasks based on their weights(ascending order).
- Based on labels sort the tasks.
- keep track of the elapsed time. We iterate through the given list of tasks and for each task we add the task duration to the elapsed time.
- As long as the elapsed time is not past the deadline, we set the task completion date to the elapsed time and add the task to a list of completed tasks. Based on need of user, we calculate hits and find probability
- Finally find average for all iterations.



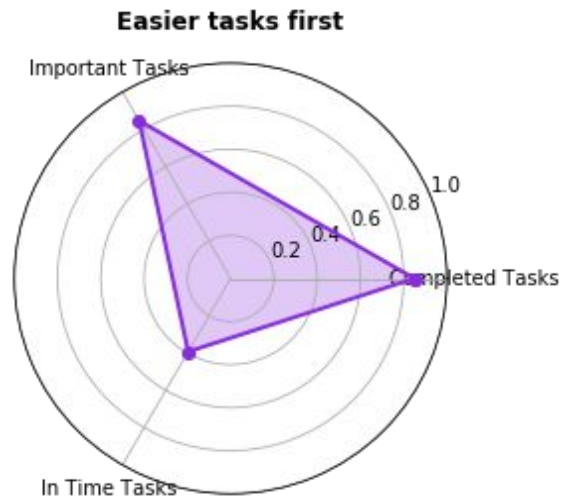
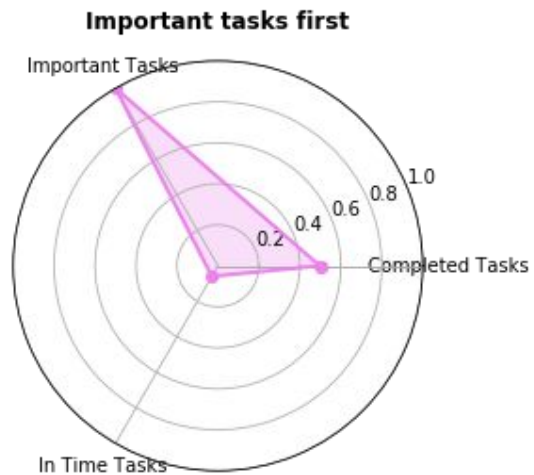
Results :

	Tasks completed in %	Important tasks completed in %	Tasks completed in time in %
Do as they come	48.79	48.33	5.86
Due tasks first	79.61	79.37	19.66
Due tasks last	18.42	18.14	3.29
Important tasks first	49.75	100	5.92
Easier tasks first	85	84.54	39.28
Easier important tasks first	79.66	94.47	39.35
Easier due tasks first	78.97	78.72	40.96

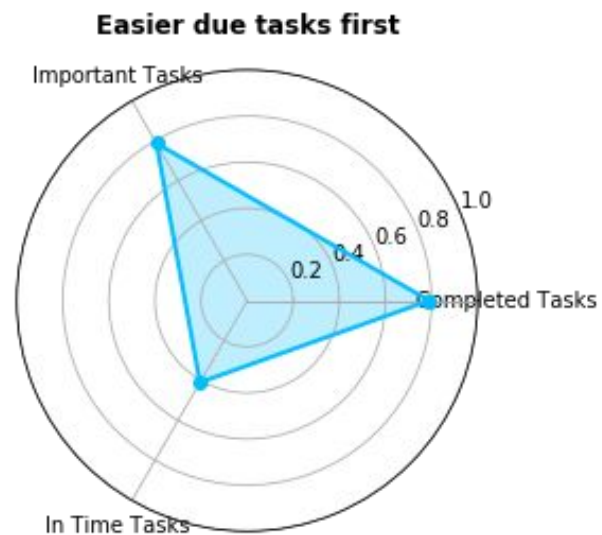
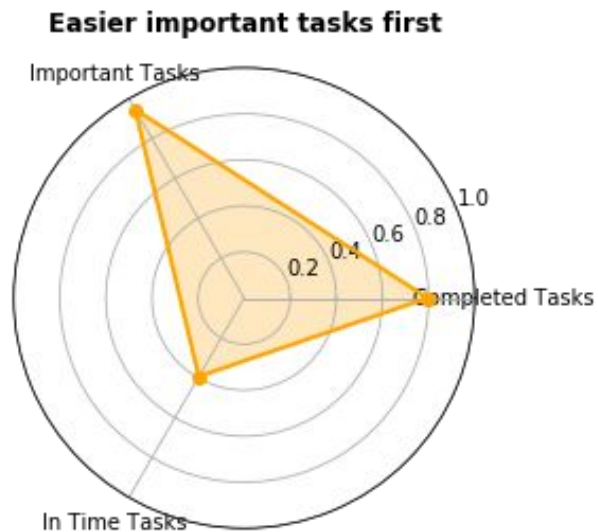
Visualization



Contd...



Contd...



Rejection sampling



Rejection Sampling

- The idea of rejection sampling is that although we cannot easily sample from f , there exists another density g , like a Normal distribution or some other from which it is easy for us to sample .
- Then we can sample from g directly and then “reject” the samples in a strategic way to make the resulting “non-rejected” samples look like they came from f .
- The density g will be referred to as the “candidate density” and f will be the “target density”.



contd..

- In order to use the rejections sampling algorithm, we must first ensure that the support of f is a subset of the support of g .
- This makes sense: if there's a region of the support of f that g can never touch, then that area will never get sampled.
- In addition, we must sample under restriction that $f(X) < kg(X)$, where $k > 1$ is appropriate bound on $f(x)/g(x)$.



Algorithm

- The rejection sampling algorithm for drawing a sample from the target density f is
 - Simulate $U \sim \text{Unif}(0,1)$.
 - Simulate a candidate $X \sim g$ from the candidate density
 - If $U \leq f(X) / kg(X)$ then “accept” the candidate X . Otherwise, “reject” X and go back to the beginning.
- The algorithm can be repeated until the desired number of samples from the target density f has been accepted.

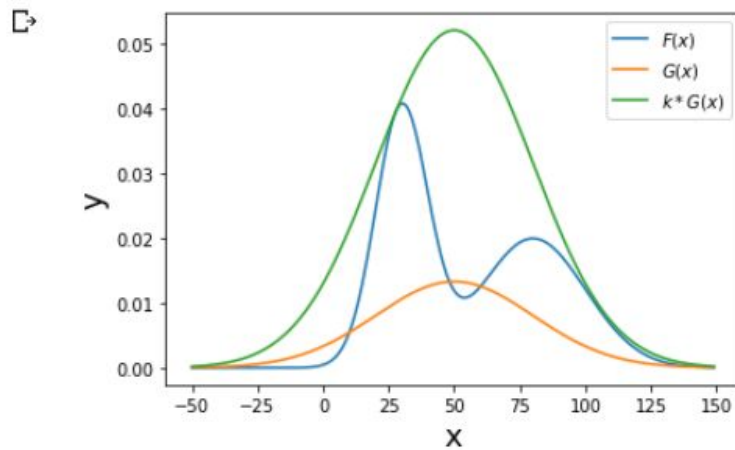


Application


Using rejection sampling to **draw samples** from a complex function.

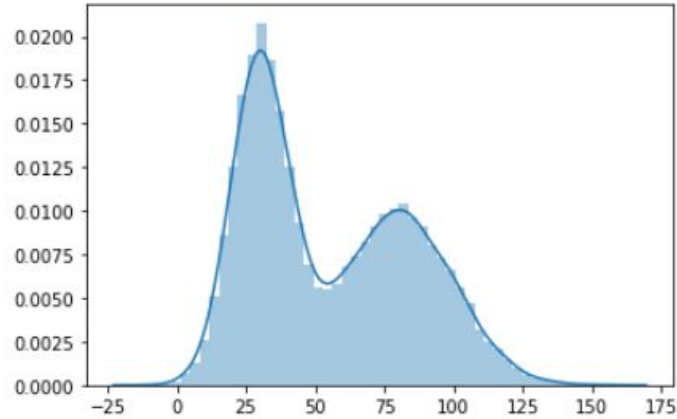
- **Number of simulations**=1000
- **The target distribution $F(x) = N(30, 10) + N(80, 20)$**
- **The (Approximated PDF) candidate distribution $G(x) = N(50, 30)$**
- **Scaling factor $k = \max(F(x) / G(x))$**

Visualization



contd..

 <matplotlib.axes._subplots.AxesSubplot at 0x7fe8deab9f28>



- From the graphs it can be seen that the accepted samples are distributed as if they were from the target distribution.

Importance sampling



Importance Sampling

- The main idea of importance sampling is to sample from a distribution q (hopefully with $q(x)$ roughly proportional to $f(x) \cdot p(x)$), and then reweigh the samples in a principled way, so that their sum still approximates the desired integral.
- More formally, suppose we are interested in computing $E_{x \sim p}[f(x)]$. We may rewrite this integral as

$$\begin{aligned} E_{x \sim p}[f(x)] &= \sum_x f(x) p(x) \\ &= \sum_x f(x) \frac{p(x)}{q(x)} q(x) \\ &= E_{x \sim q}[f(x) w(x)] \end{aligned}$$



contd..

- Where, $w(x)=p(x)/q(x)$ and the samples x are drawn from q . In other words, we may instead take samples from q and reweigh them with $w(x)$; the expected value of this Monte Carlo approximation will be the original integral.

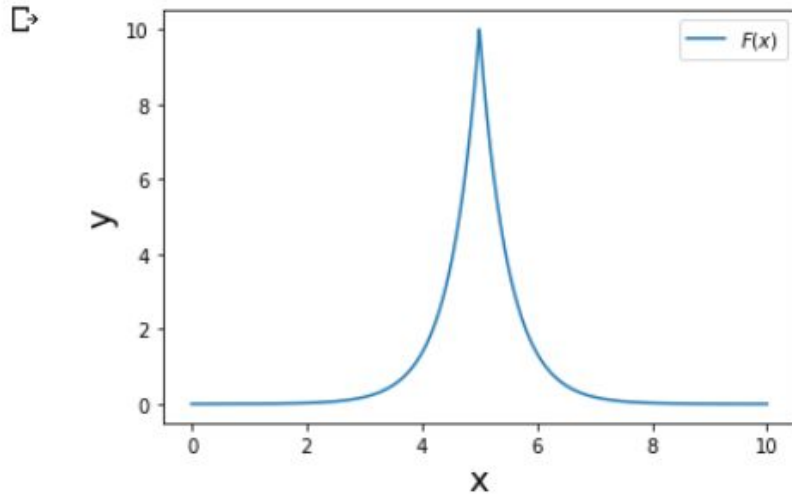


Application

Using importance sampling here provides a **reduction in the variance** of an integral approximation.

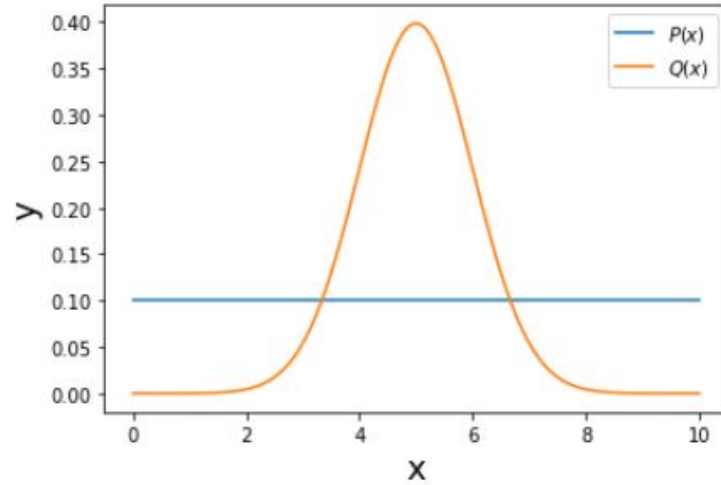
- **Number of simulations**=1000
- **F(x)** = $10e^{-2|x - 5|}$
- **P(x)** = $1/10$ (uniform)
- **Q(x)** = $N(5, 1)$

Visualization



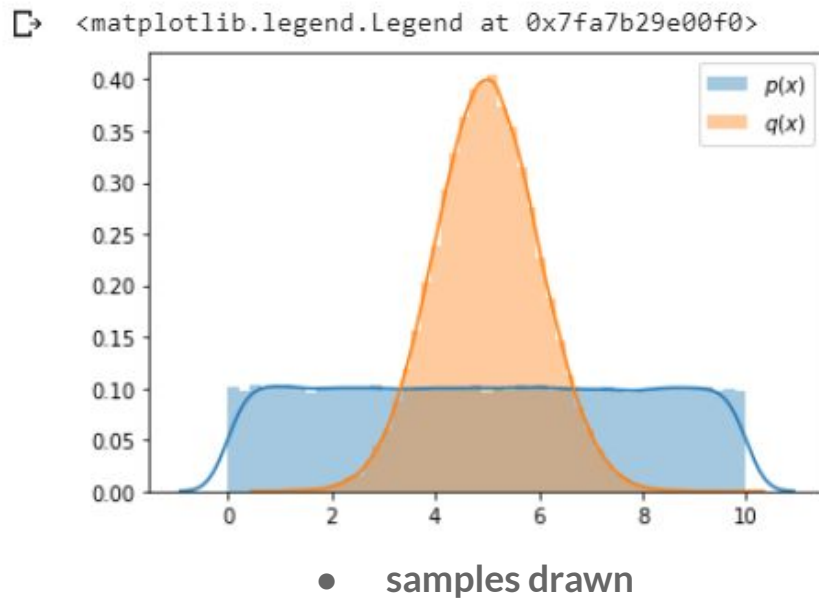
- Function to integrate

contd..



- functions taken

contd..





Results

actual integration value : 9.999545984033862

Sampling From $p(x)$:

$\mathbf{x}f(\mathbf{x})p(\mathbf{x}) = 0.9954587219729663$

average 0.9954587219729663 variance 3.9950488797098678

Approximated Integral value from $p(\mathbf{x})$: 9.954587219729662

Diff from actual = 0.04495876430419976



contd..

Importance sampling Result:

`xf(x)w(x)q(x) = 0.9978580875986263`

`average 0.9978580875986263 variance 0.35524179879268936`

`Approximated Integral value from importance sampling: 9.978580875986262`

`Diff from actual = 0.020965108047599657`

The integral calculation is still correct (and nearly same), but with a variance this is approximately 1/10 of the simple monte carlo integral approximation

Markov Chain



Markov Chains

- A Markov chain is a stochastic model describing a sequence of possible events in which the probability of each event depends only on the state attained in the previous event.
- Markov Chain is used to model random real world processes as mathematical models.
- Example : Currency Exchange Rates etc



Application of Markov Chain

Application : Predicting Next word in a text from the previous given text piece

Methodology :

- Here, we have taken Donald Trump's Speech during his campaigning for elections.
- The words used by him in his speech are stored and a transition matrix is formed for the all probabilities.



Continued...

Methodology :

- For the given text(input), the last word is taken, and the conditional probabilities are used to calculate to predict next word.
- From the obtained probabilities, we are taking the word with maximum probability as the next word and getting the next required word in the same way.



Continued...

Data : Sample(part) of the speech used

It will be American steel that sends our skyscrapers soaring into the sky.
It will be American steel that rebuilds our inner cities.
It will be American hands that remake this country, and it will be American energy -
It will be American workers who are hired to do the job.
We are going to put American-produced steel back into the backbone of our country. T
On trade, on immigration, on foreign policy, we are going to put America First again
We are going to make America wealthy again.
We are going to reject Hillary Clinton's politics of fear, futility, and incompetenc
We are going to embrace the possibilities of change.
It is time to believe in the future.
It is time to believe in each other.
It is time to Believe In America.
This Is How We Are Going To Make America Great Again - For All Americans.
We Are Going To Make America Great Again For Everyone - Greater Than Ever Before.
Thank you.



Results

Output Generated :

"L.A. -- TRADE DEALS. JUST CAME OUT IN IRAQ. BUT THEY ARE LIKE HIS LEG, RIGHT. YOU'RE NOT REALLY REPRESENTING YOUR FOOTBALL TEAM -- ONE OF US, I'M LOOKING AT THIS PERSON SAID KEEP OUR MILITARY. OUR ENTIRE NATION GRIEVES AND THEY LET ME TO THE REASON I think it's Trump"

Markov Chain Monte Carlo Methods



Markov Chain Monte Carlo Methods

Markov chain Monte Carlo (MCMC) methods comprise a class of algorithms for sampling from a probability distribution. By constructing a Markov chain that has the desired distribution as its equilibrium distribution, one can obtain a sample of the desired distribution by recording states from the chain.

There are many types of MCMC Methods:

- Metropolis-Hastings
- Simulated Annealing
- Gibbs Sampling



Metropolis-Hastings

The **Metropolis-Hastings algorithm** is a Markov chain Monte Carlo (MCMC) method for obtaining a sequence of random samples from a probability distribution from which direct sampling is difficult.

Algorithm :

- Suppose we have a target distribution π ,
- And a transition kernel(matrix) Q .
- Initialize $X_1 = x_t$



Continued...

- For $t = 1, 2, \dots$
 - sample y from $Q(y|x_t)$. Think of y as a “proposed” value for x_{t+1} .
 - Compute

$$A = \min \left(1, \frac{\pi(y)Q(x_t|y)}{\pi(x_t)Q(y|x_t)} \right).$$

A is often called the “acceptance probability”.

- with probability A “accept” the proposed value, and set $x_{t+1} = y$. Otherwise set $x_{t+1} = x_t$.



Continued...

- Generate a uniform Random Number u in $[0,1]$ and

Accept if $A \geq u$, $x(t)$, and $x(t+1) = x_1$

Reject if $A < u$ and $x(t+1) = x(t)$



Application :

Application : Decrypting a given encrypted text with finding deciphering key

Assumption :

- Score function is used to calculate acceptance probability.

Methodology :

- Picks up a random current state(decryption cipher)
- New state proposal by swapping any 2 letters in the current state.

Continued...

- Calculates the score of the new state(using scoring function) and compares it with the previous state.

- $$Score(x) = \prod R(\beta_1, \beta_2)^{F_x(\beta_1, \beta_2)}$$

- if `score(current_state) < score(proposed_state)`:

`current_State = proposed_state`

Else :

`u[0,1] >= score(proposed_state)/score(current_state)`

(u is random number from Uniform Distribution)

If `flip == head` :

`Current_state = proposed_State`

Else :



Continued...

Data : The data used here is War and Peace novel text by Tolstoy

Input : Text :

```
Text To Decode: XZ STAVRK HXVR MYAZ OAKZM JKSSO SO MYR OKRR XDP JKSJRK XBMA5D SO YAZ TWDHZ MYR JXMBYNSKF BSVRKTRM NYABY NXZ BX
KRTRZZTQ OTWDH SVRK MYR AKSD ERPZMRXP KWZMTRP MYR JXTR OXBR SO X Q5WDH NSIXD NXZ KXAZRP ORRETQ OKSI MYR JATT5N XDP X OXADM V
SABR AIJRKOR5MTQ XKMABWTXMRP MYR NSKPZ TRM IR ZRR MYR BYATP XDP PAR MYR ZWKHRSD YXP ER5D ZAMMADH NAMY YAZ OXBR MWKDRP MSNXKP
Z MYR OAKR HAVADH MYR JXTIZ SO YAZ YXDPZ X NXKI XDP X KWE X5MRKDXMRTQ XZ MYR Q5WDH NSIXD ZJSFR YR KSZR XDP XPVXDBADH MS MYR
ERP Z YRXP ZXAP NAMY ISKR FAD5DRZZ MYXD IAHYM YXVR ER5D RGJRBMRP SO YAI
```



Iterations :

```
iter 0 : XZ STAVRK JXVR MYAZ OAKZM HKSSO SO MYR OKRR XDP HKSHRK XBMSD SO YAZ TWDJZ MYR HXMBYNSKF BSVRKTMR
iter 500 : TN OIAHER DTHE SPAN FARNB BROOF OF SPE FREE TLY BROBER TUSAOL OF PAN IGLDN SPE BTSUPCORV UOHERIES
iter 1000 : AS OIUPER YAPE THUS FURST BROOF OF THE FREE ALD BROBER ANTUOL OF HUS IGLYS THE BATNHCORK NOPERIET
iter 1500 : AS OLIBER GABE THIS FIRST CROOF OF THE FREE AND CROCEB APTION OF HIS LKNGS THE CATPHMORM POBERLET
iter 2000 : AS OLIVER GAVE THIS FIRST CROOF OF THE FREE AND CROCEB APTION OF HIS LKNGS THE CATPHWORM POVERLET
iter 2500 : AS OLIVER GAVE THIS FIRST PROOF OF THE FREE AND PROPER ACTION OF HIS LUNGS THE PATCHWORK COVERLET
iter 3000 : AS OLIVER GAVE THIS FIRST PROOF OF THE FREE AND PROPER ACTION OF HIS LUNGS THE PATCHWORK COVERLET
iter 3500 : AS OLIVER GAVE THIS FIRST PROOF OF THE FREE AND PROPER ACTION OF HIS LUNGS THE PATCHWORK COVERLET
iter 4000 : AS OLIVER GAVE THIS FIRST PROOF OF THE FREE AND PROPER ACTION OF HIS LUNGS THE PATCHWORK COVERLET
iter 4500 : AS OLIVER GAVE THIS FIRST PROOF OF THE FREE AND PROPER ACTION OF HIS LUNGS THE PATCHWORK COVERLET
iter 5000 : AS OLIVER GAVE THIS FIRST PROOF OF THE FREE AND PROPER ACTION OF HIS LUNGS THE PATCHWORK COVERLET
iter 5500 : AS OLIVER GAVE THIS FIRST PROOF OF THE FREE AND PROPER ACTION OF HIS LUNGS THE PATCHWORK COVERLET
iter 6000 : AS OLIVER GAVE THIS FIRST PROOF OF THE FREE AND PROPER ACTION OF HIS LUNGS THE PATCHWORK COVERLET
iter 6500 : AS OLIVER GAVE THIS FIRST PROOF OF THE FREE AND PROPER ACTION OF HIS LUNGS THE PATCHWORK COVERLET
iter 7000 : AS OLIVER GAVE THIS FIRST PROOF OF THE FREE AND PROPER ACTION OF HIS LUNGS THE PATCHWORK COVERLET
iter 7500 : AS OLIVER GAVE THIS FIRST PROOF OF THE FREE AND PROPER ACTION OF HIS LUNGS THE PATCHWORK COVERLET
iter 8000 : AS OLIVER GAVE THIS FIRST PROOF OF THE FREE AND PROPER ACTION OF HIS LUNGS THE PATCHWORK COVERLET
iter 8500 : AS OLIVER GAVE THIS FIRST PROOF OF THE FREE AND PROPER ACTION OF HIS LUNGS THE PATCHWORK COVERLET
iter 9000 : AS OLIVER GAVE THIS FIRST PROOF OF THE FREE AND PROPER ACTION OF HIS LUNGS THE PATCHWORK COVERLET
iter 9500 : AS OLIVER GAVE THIS FIRST PROOF OF THE FREE AND PROPER ACTION OF HIS LUNGS THE PATCHWORK COVERLET
```



Decoded Text :

Decoded Text: AS OLIVER GAVE THIS FIRST PROOF OF THE FREE AND PROPER ACTION OF HIS LUNGS THE PATCHWORK COVERLET WHICH WAS CARELESSLY FLUNG OVER THE IRON BEDSTEAD RUSTLED THE PALE FACE OF A YOUNG WOMAN WAS RAISED FEEBLY FROM THE PILLOW AND A FAINT VOICE IMPERFECTLY ARTICULATED THE WORDS LET ME SEE THE CHILD AND DIE THE SURGEON HAD BEEN SITTING WITH HIS FACE TURNED TOWARDS THE FIRE GIVING THE PALMS OF HIS HANDS A WARM AND A RUB ALTERNATELY AS THE YOUNG WOMAN SPOKE HE ROSE AND ADVANCING TO THE BED S HEAD SAID WITH MORE KINDNESS THAN MIGHT HAVE BEEN EXPECTED OF HIM

Decrypted Key :

MCMC KEY FOUND: ICZNBKXGMPRJTWFDYEOLQVUAHS
ACTUAL DECRYPTION KEY: ICZNBKXGMPRQTWFDYEOLJVUAHS

Simulated Annealing



Simulated Annealing

- Simulated Annealing is an optimization technique among the MCMC Methods.
- It is used for solving unconstrained and bound constrained optimization problems.
- It is majorly used in Machine Learning problems.
- It is an alternative for Gradient Descent.
- At each step, the simulated annealing heuristic considers some neighboring state s^* of the current state s , and probabilistically decides between moving the system to state s^* or staying in-state s .



contd..

- These probabilities ultimately lead the system to move to states of lower energy.
- Typically this step is repeated until the system reaches a state that is good enough for the application, or until a given computation budget has been exhausted.



Application

- We used Simulated Annealing to solve the knapsack problem.
- In the knapsack problem, we have to find the maximum gold digger can take based on the gold value and weight of the bag from N number of bags of different weights and different gold values.
- The steps involved are :
 - Step 1: Create two different arrays with gold values and weights. Let w_{max} be the maximum weight taken by the gold digger.



contd..

- Step 2 : Pick a random state from the array and toggle the index value.
- Step 3 : Check if we satisfy our constraint. If yes this state is the proposal state.(Our constraint is the scoring function if the person can take the weight and it has the maximum gold value)

- Step 4 :

$$Score(X) = e^{\beta GX^T}$$

- Here, beta is a positive constant, G is the gold value of the selected state and X is the weight of the selected state. The more the score, the preferable the bag is. X should be less than wmax.
- This optimization is carried out by Simulated Annealing.

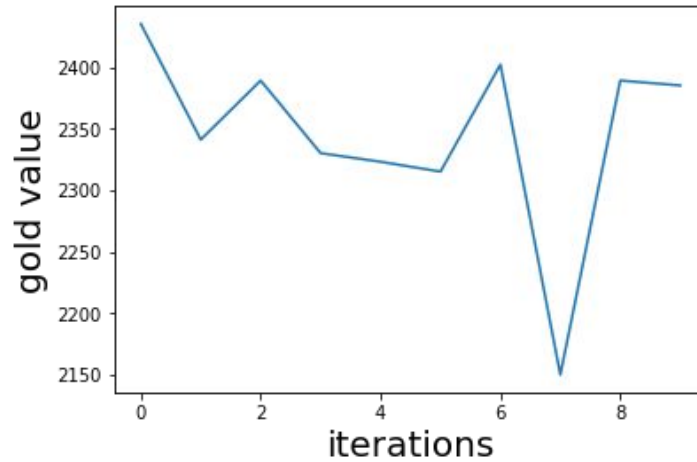


Visualization

```
W = [20, 40, 60, 12, 34, 45, 67, 33, 23, 12, 34, 56, 23, 56]  
G = [120, 420, 610, 112, 341, 435, 657, 363, 273, 812, 534, 356, 223, 516]  
W_max = 150
```

- The weight and gold value arrays for the problem

contd..



- Solutions at each iteration



Results

From the graph we can observe that the maximum gold value is 2435 and we got the following array as output.

The array here represents the gold values that are taken.

```
MCMC Solution is : [0, 0, 0, 1, 1, 0, 0, 1, 1, 1, 1, 0, 0, 0] with Gold Value: 2435
```

Gold Bag index representing whether the bag is among the taken or not. X Label is the gold bag index.

Gibbs Sampling



Gibbs sampling

- Gibbs Sampling is a MCMC method to draw samples from a potentially really really complicated, high dimensional distribution, where analytically, it's hard to draw samples from it.
- Generally gibbs sampling is as follows:
 1. Set initial values for all parameters $\Theta_1, \dots, \Theta_p$
 2. Variables are sampled one at a time from their conditional distributions i.e,
$$p(\Theta_j | \Theta_1, \dots, \Theta_{j-1}, \Theta_{j+1}, \dots, \Theta_p, y)$$
 3. Process is repeated until a required number of samples are generated.



Application - Spam filter

- Estimate the prevalence(Ψ) of spam without directly counting instances of spam.
- $S=1$ if email in fact is spam. $S=0$ if email is not spam.
- $R=1$ if email is marked as spam and $R=0$ if email is marked as not spam.
- Terminology
 - Sensitivity $\eta=P(R=1|S=1)$
 - Specificity $\theta=P(R=0|S=0)$
 - Prevalence $\Psi=P(S=1)$



Contd...

- Assumptions
 - $\eta=0.9, \Theta=0.95$, rejected emails(r)=233, Burn-in=2000, $\Psi[0]=0.5$
 - $\alpha=1, \beta=1$,
 - total number of emails(n)=1000.
- Formulas used

$$\tau = P(R=1) = P(R=1, S=1) + P(R=1, S=0) = \Psi\eta + (1 - \Psi)(1 - \Theta)$$

Finally we get $\Psi = (\tau + \Theta - 1)/(\eta + \Theta - 1)$

$$p = P(S=1|R=1) = (\eta\Psi)/\tau$$

$$q = P(S=1|R=0) = (\Psi(1 - \eta))/(1 - \tau)$$



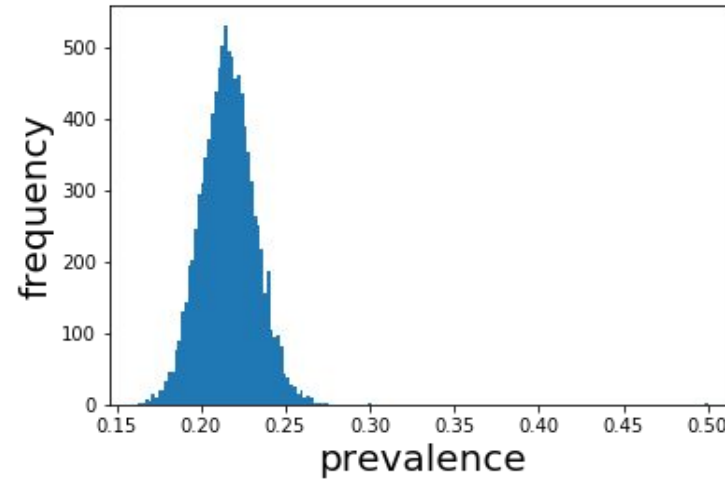
Algorithm

- X be number among r rejected emails that are spam . Y be number of spam emails among $(n-r)$ passes emails.
- For i in number of iterations:
 1. Calculate τ ,
 2. Randomly generate
$$X|_{r, \Psi} \sim \text{Binorm}(r, p)$$
$$Y|_{r, \Psi} \sim \text{Binorm}(n-r, q)$$
$$\Psi | X, Y \sim \text{Beta}(\alpha + X + Y, \beta + n - X - Y)$$

Find mean of Ψ leaving first 2000 samples because our burn in number is 2000

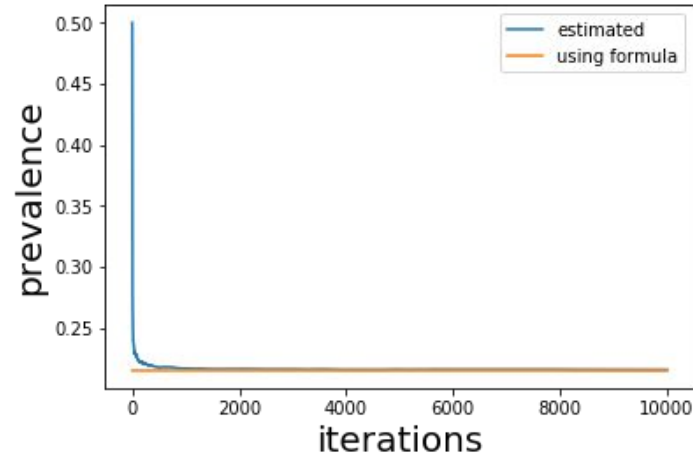
Results

- Histogram of prevalence Ψ :



Contd...

- Iterations vs prevalence



- From the above results we can find that using gibbs sampling estimate prevalence converges to prevalence calculated using formula(0.215).That is 20% of emails are in fact spam.

Thank You

