Reminders:

- Assignment 2, due today. (Secant Method and 2D Newton-Raphson)
- Art show
-

Last week: Lab Task #1

- Code up the Urn model.
- Start with 1 red and one blue ball
- Continue until you have 50 balls
  1. What is the expected number of red balls at the end? (Intuition?)
  2. What is the distribution of the number of red balls? (Plot a histogram - what properties do you think the distribution will have?)
- Suggest: Use at least 1000 replicates.
- We will then look at the results.
  - Empirical estimate of expected value of a distribution is just the mean of the observed values (i.e. mean number of red balls across the 1000(say) reps.)
  - Empirical estimate of a distribution is the histogram.

Last week: Lab Task #2

- Code up the Urn model
- Start with any number, n, of red and blue balls
- Assume n-1 of the starting balls are red (and so 1 is blue)
- Continue until you have 50 balls

1. What is the expected number of red balls at the end, as a function of the number we start with? (Draw a plot of mean # of reds, versus #reds at start). (Also draw a plot of mean #reds versus proportion of reds at start.)
2. Is this described by a simple relationship?

– So, try using a fixed number of balls at the end (say 50), but with different numbers of starting balls (2,3,4,...,10, say)

– Compare expected number of red balls as end to:

  – Number of red balls at start;

  – Proportion of red balls at start.

Lab Task #4 - The Polya Urn

- Code up this Polya Urn model. Then try the following:
- Start with one 'red' ball and the black ball (but code the colors as numbers, for simplicity's sake, so Ball1<-1, Ball2<-2, say).
- Repeat until you have 50 non-black balls; replicate this process multiple times.

1. What is the distribution of the number of (non-black) colors at the end?
2. What is the distribution of the number of balls of the commonest color at the end?
3. How many replicates do you need to do to answer these questions? How did you decide upon this number?
4. Repeat 1. and 2., but start with two balls of a same color+black ball each time. Compare your results to those from 1. and 2. Can you explain what you see?

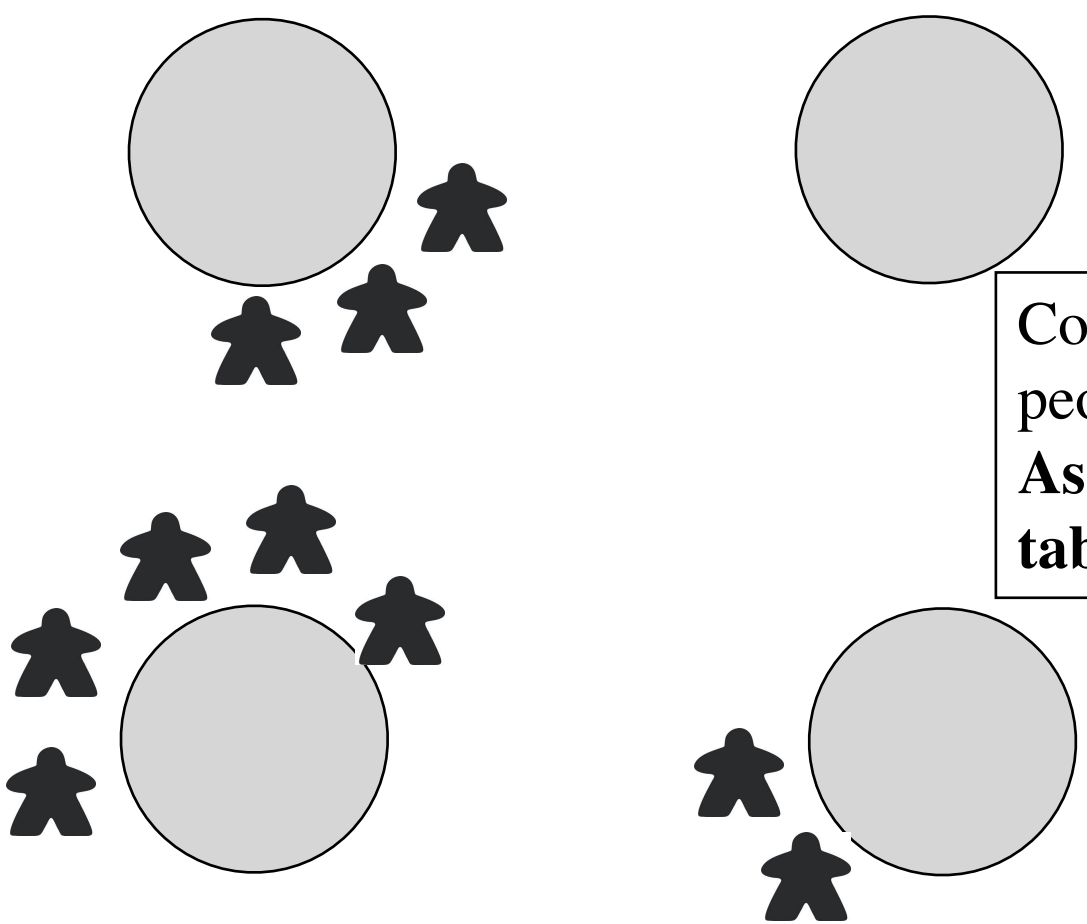
Lab Task 5 - The Generalized Polya Urn (note: you will need this version later in the course, so it better work!)

- Code up a further generalized version of the Polya Urn model
  - Imagine that each color has a 'weight'
  - When we draw a ball, a given ball is chosen with probability equal to (Weight of that ball)/(total weight of all balls in the Urn).
  - [So, our previous version had implicit weights of 1 for each color]
- Start with two 'red' balls and the black ball (but, again, code the colors as numbers, for simplicity's sake).
- Repeat until you have 50 non-black balls; replicate this process multiple times.
- Assume all, non-black colors have weight 1
  1. What is the expected number of different (non-black) colors in the Urn at the end as a function of the weight of the black ball ?
  2. What is the distribution of the number of (non-black) colors at the end, as a function of the weight of the black ball? (Try weight=1, or 2, or, ... , or 10).

The Restaurant Process

Customer n+1 does one of two things:

1. With probability 1/(n+1) they sit at an unoccupied table
2. Otherwise, they picksa person, uniformly at random, and sits at the same table, to their right.



Continue until we have n people seated.

Assume the supply of tables is unlimited.

Task - the Restaurant Process [RP] 1

- Write a function to simulate the RP.
- Write another function to replicate the RP many times.
- Suppose we eventually have 50 customers.
  - What is the expected number of occupied tables?
  - What is the distribution of the number of occupied tables?

Task - the Restaurant Process 2

- Suppose that, conditional on there currently being N customers already seated, the prob. that the next customer sits at a new table is $\theta/(N+\theta)$, rather than $1/(N+1)$

- Suppose we eventually have 50 customers.
  - What is the expected number of occupied tables as a function of θ? [Suggest try θ=0.1,0.5,1,2,3,4,...,10,100]
  - How does the distribution of the number of occupied tables vary as a function of θ?
  - How does the run time vary as a function of θ?

- Do you see any connection with your answers to Urn Task 5 here, or between Urn Task 4 and the previous problem?

Monte Carlo Simulation: Monty Hall Problem

- Game show ("Let's Make a Deal") with 3 doors
- Behind 1 door is $$$ (a fast car, a lifetime's supply of chocolate, or some other source of eternal happiness)
- Behind the other 2 doors are goats (sheep, donkey, undesirable 'prize' of choice)
- NB. goats are bound, gagged and de-odorized, so none of your senses (except common sense) will help you.

How it works

- You pick a door.
- Before he opens that door, Monty opens one of the other doors, revealing a goat.
- He then offers you a chance to switch to the other un-opened door.
- Question: Should you switch doors?

- It's an easy thing to simulate...

Monty Hall Problem

- http://math.ucsd.edu/~crypto/Monty/monty.html
- "Let's make a deal" website

Journal of Comparative Psychology
2010, Vol. 124, No. 1, 1–13

0735-7036/10/$12.00 DOI: 10.1037/a0017703

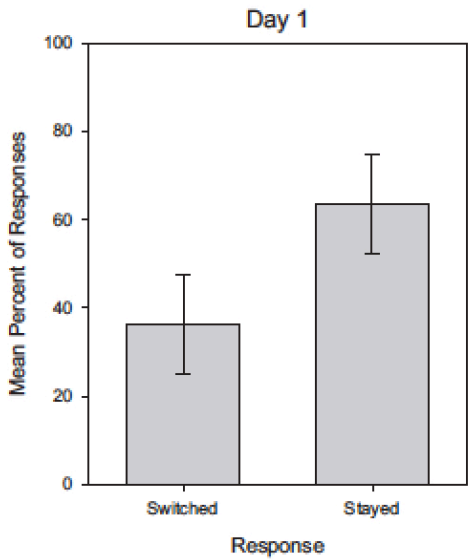
Are Birds Smarter Than Mathematicians? Pigeons (*Columba livia*) Perform Optimally on a Version of the Monty Hall Dilemma

Walter T. Herbranson and Julia Schroeder
Whitman College

The "Monty Hall Dilemma" (MHD) is a well known probability puzzle in which a player tries to guess which of three doors conceals a desirable prize. After an initial choice is made, one of the remaining doors is opened, revealing no prize. The player is then given the option of staying with their initial guess or switching to the other unopened door. Most people opt to stay with their initial guess, despite the fact that switching doubles the probability of winning. A series of experiments investigated whether pigeons (*Columba livia*), like most humans, would fail to maximize their expected winnings in a version of the MHD. Birds completed multiple trials of a standard MHD, with the three response keys in an operant chamber serving as the three doors and access to mixed grain as the prize. Across experiments, the probability of gaining reinforcement for switching and staying was manipulated, and birds adjusted their probability of switching and staying to approximate the optimal strategy. Replication of the procedure with human participants showed that humans failed to adopt optimal strategies, even with extensive training.

Keywords: pigeon, probability, matching, Monty Hall Dilemma

Birds



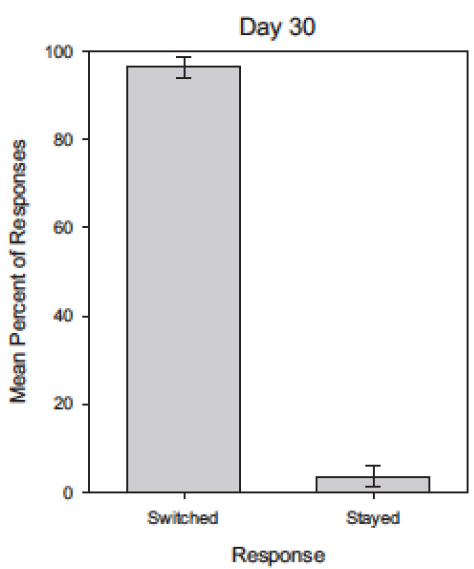
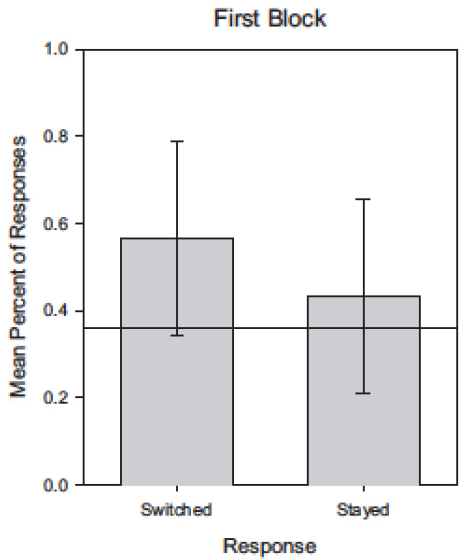


Figure 1. Proportion of switching (left bars) and staying (right bars) responses, averaged across all birds in Experiment 1. The top panel represents the first day of training, and the bottom panel represents the 30th day of training. Error bars represent 95% confidence intervals.

Humans



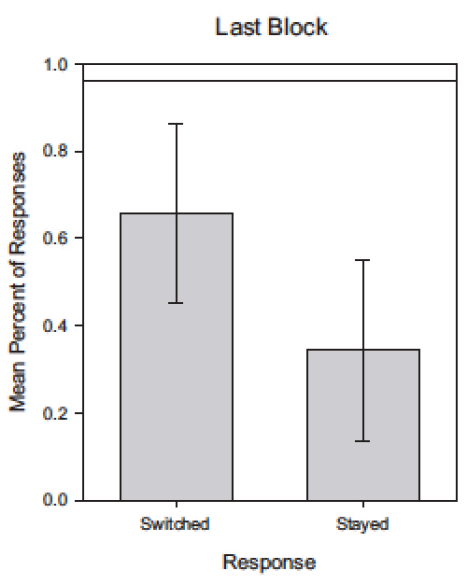


Figure 4. Proportion of switching (left bars) and staying (right bars) responses, averaged across all human participants in Condition 1 of Experiment 3. The top panel represents the first 50 trials, and the bottom panel represents the final 50 trials. Error bars represent 95% confidence intervals. Reference lines show the probability of switching by pigeons during the first (top panel) and final (bottom panel) days of training in Experiment 1.

PM520 - Lecture 5
Optimization methods

Chapter 12 of Owen, Maillardet and Robinson (both editions)

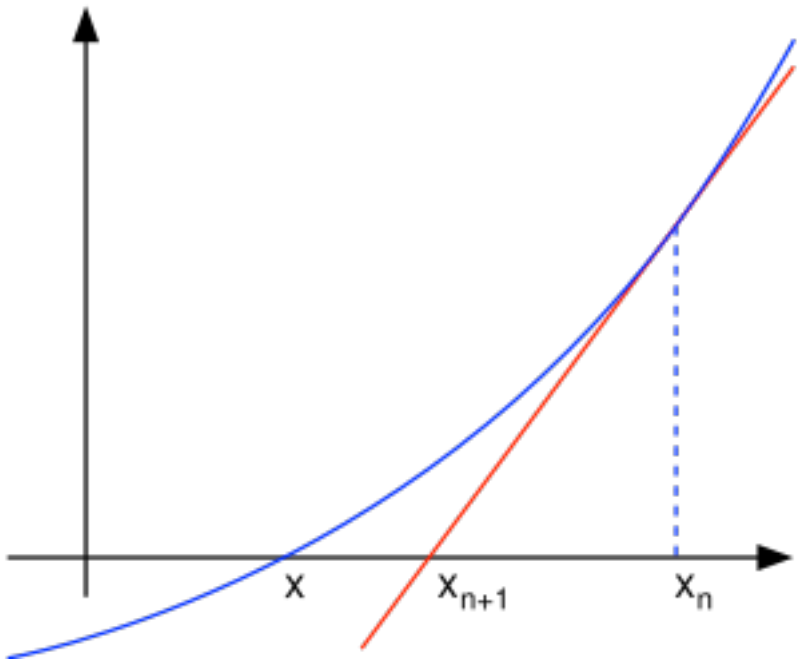
Optimization algorithms

- Explore a surface, V, looking for the maximum (or minimum) value of a function f.
  - e.g. minimize $f(x)=(x-2)^2+\ln(x)$, V=real line.
- There are a great many algorithms...

Newton's method

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}$$

- Newton-Raphson to find roots
- Start at an arbitrary value
- Sequence $\{x_n\}$ converges to a root



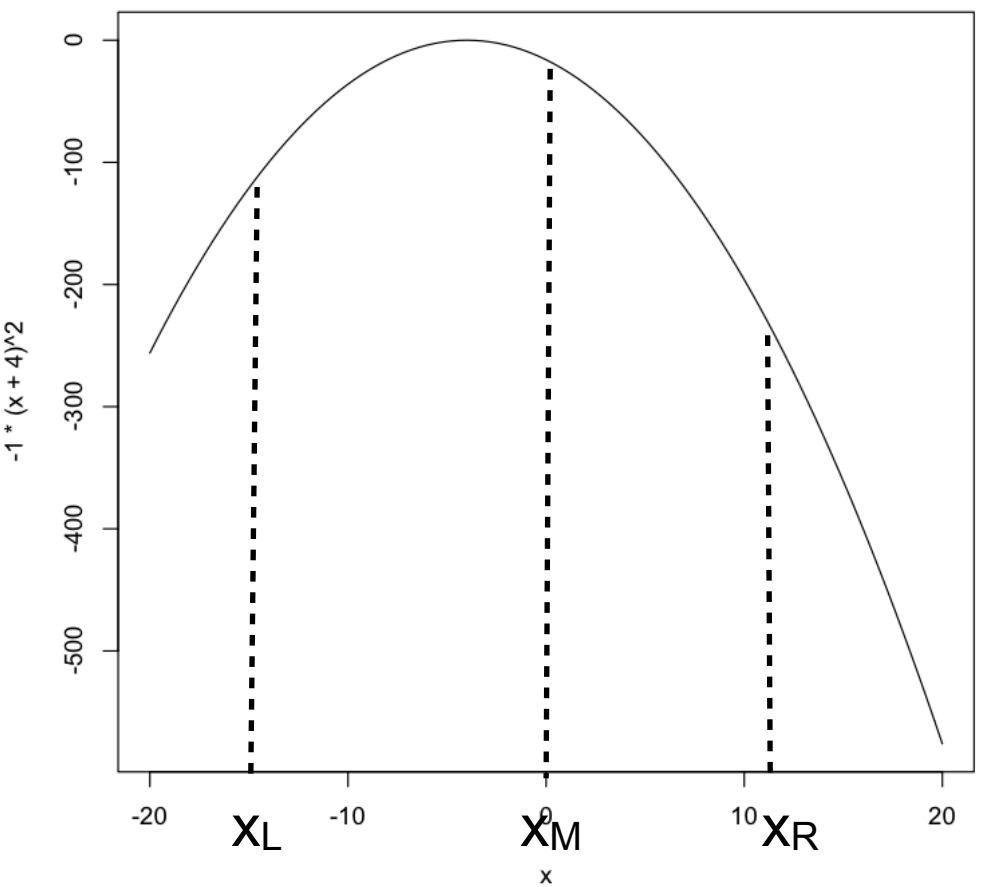
An illustration of one iteration of Newton's method (the function *f* is shown in blue and the tangent line is in red). We see that x_{n+1} is a better approximation than x_n for the root x of the function f.

- NB. Maximum (or Minimum) occurs at the root of the derivative
- So, apply Newton Raphson to f'(x), the derivative of f(x)
- Second derivative +ve -> minimum
- Second derivative -ve -> maximum

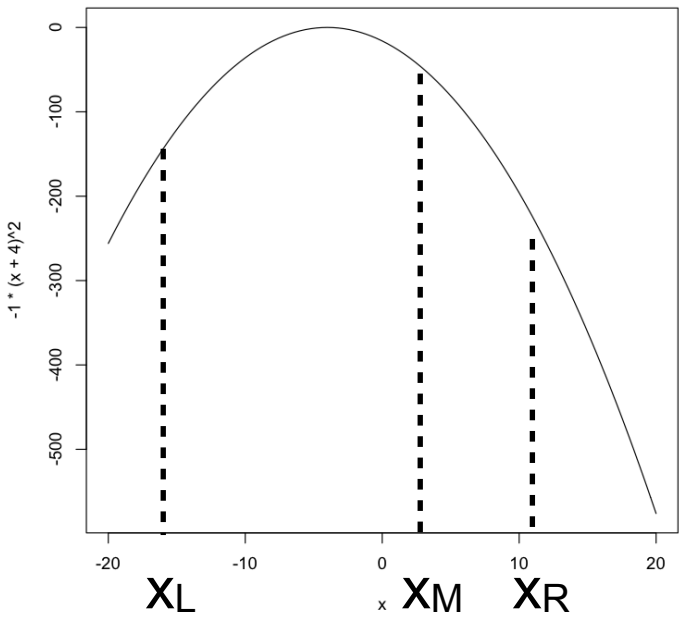
Golden section method

- Suppose we wish to find a maximum of f(), but derivatives are not available?
- Start with three values, x_L, x_M, x_R such that $f(x_L) \leq f(x_M) \geq f(x_R)$

If f is continuous, there must be a maximum between x_L and x_R



Golden section method



Pick $\varepsilon > 0$. Start with x_L, x_M, x_R such that

$f(x_L) \le f(x_M) \ge f(x_R)$

1. If $x_R - x_L \le \varepsilon$ stop.
2. If $x_R - x_M > x_M - x_L$ then do 2a;

   otherwise do 2b.

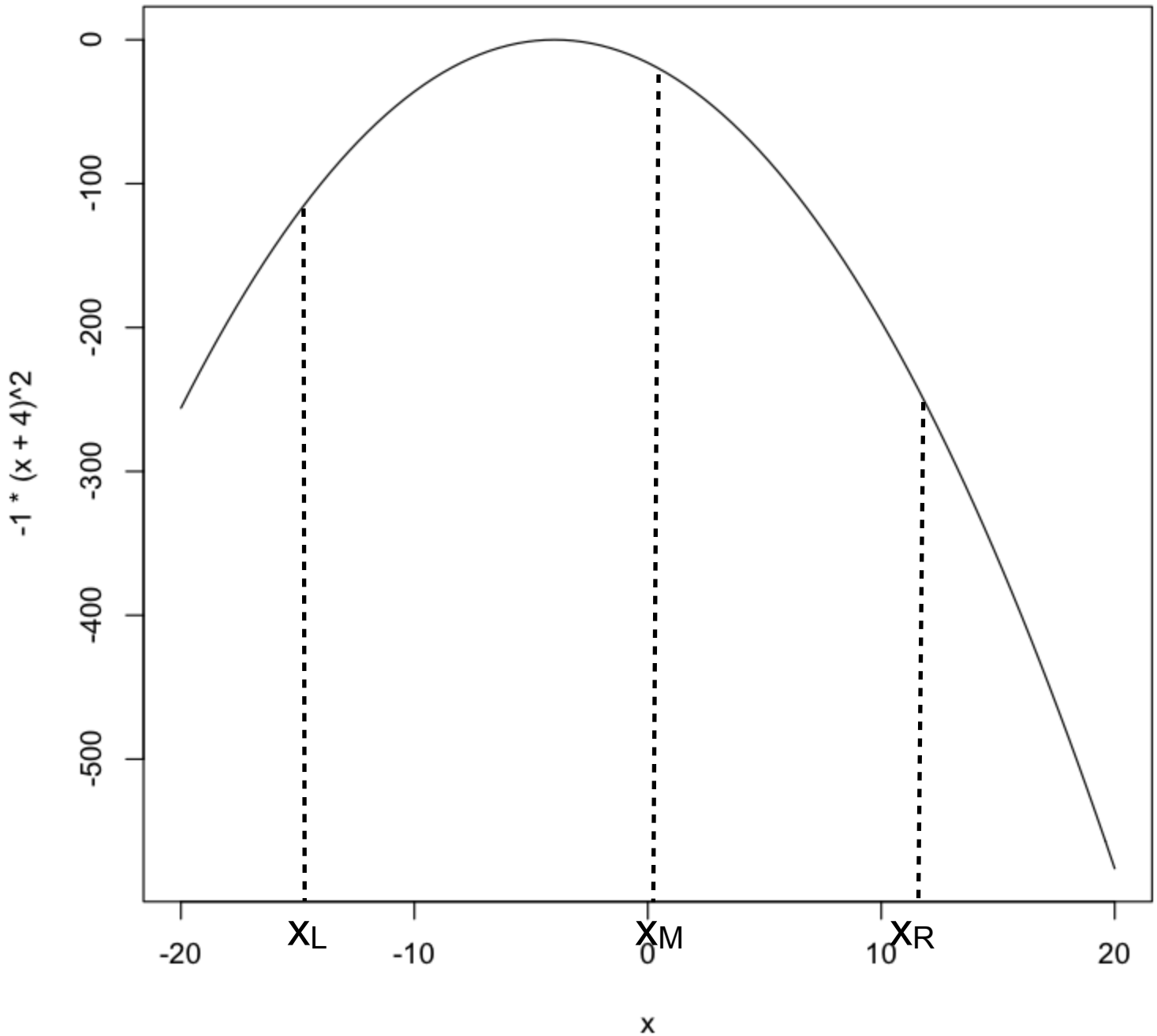
   2a. Choose a $y \in (x_M, x_R)$.

   If $f(y) \ge f(x_M)$ then put $x_L = x_M$ and $x_M = y$; otherwise put $x_R = y$.

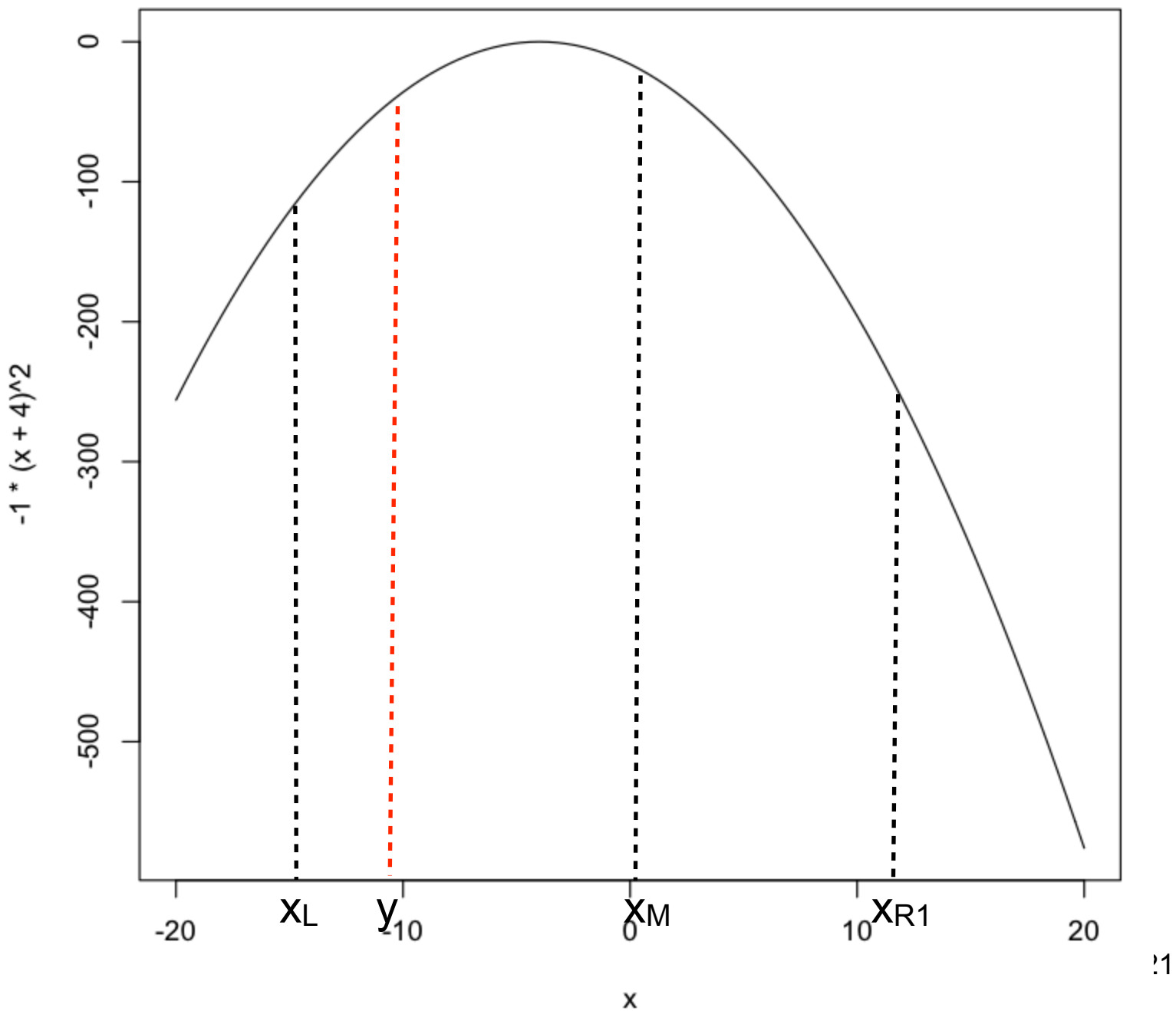
   2b. Choose a $y \in (x_L, x_M)$.

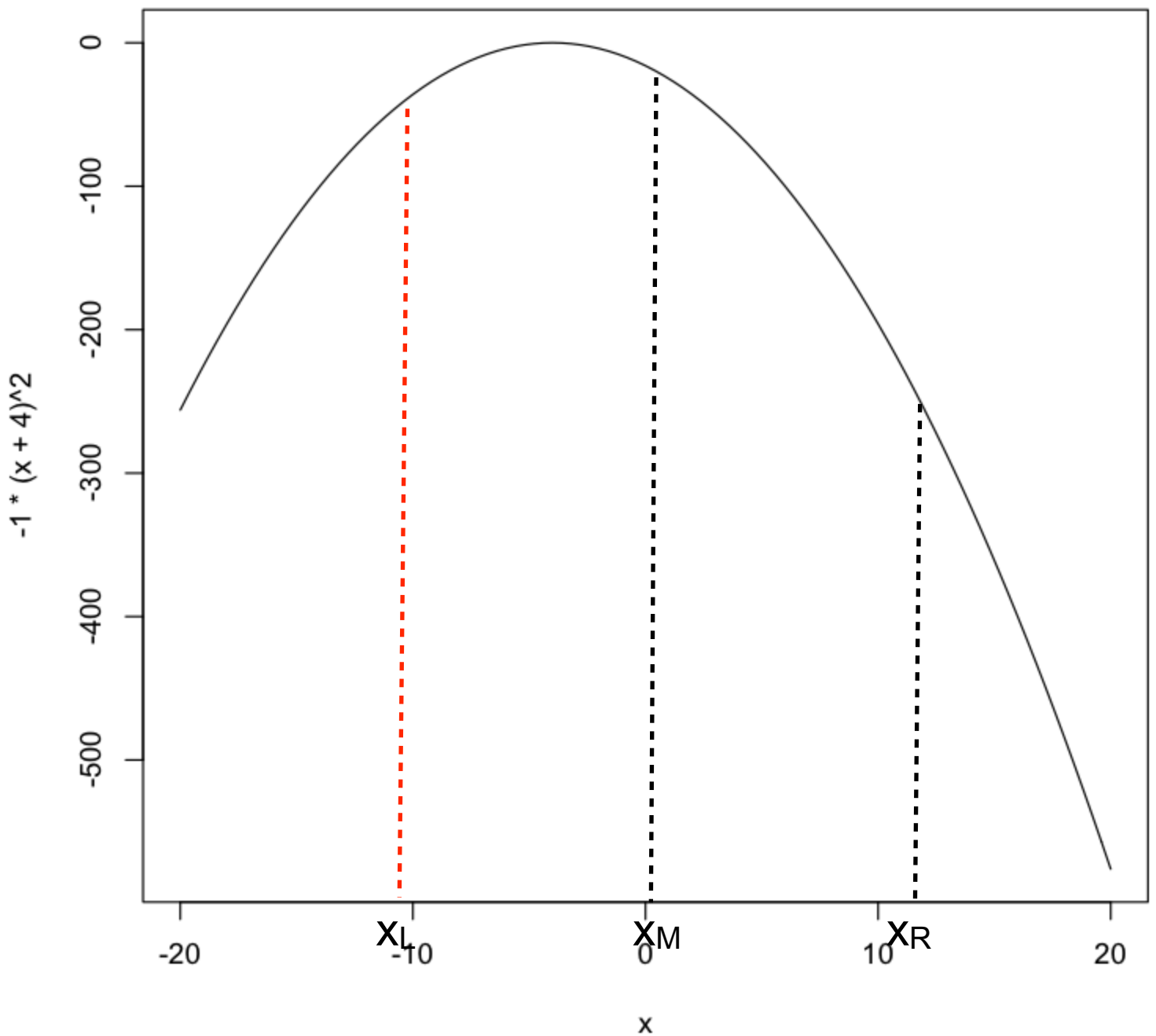
   If $f(y) \ge f(x_M)$ then put $x_R = x_M$ and $x_M = y$; otherwise put $x_L = y$.

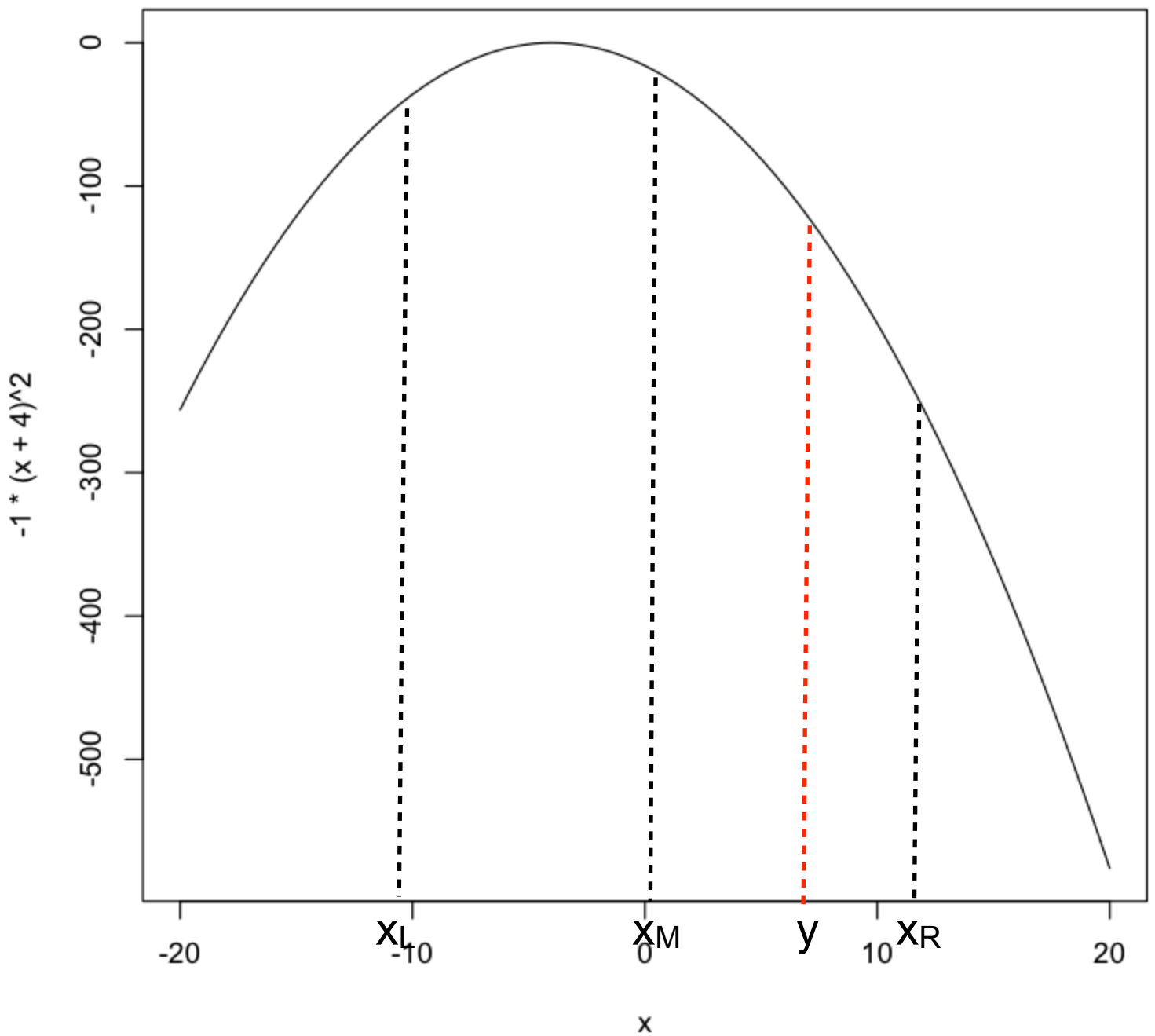
3. Go back to step 1.

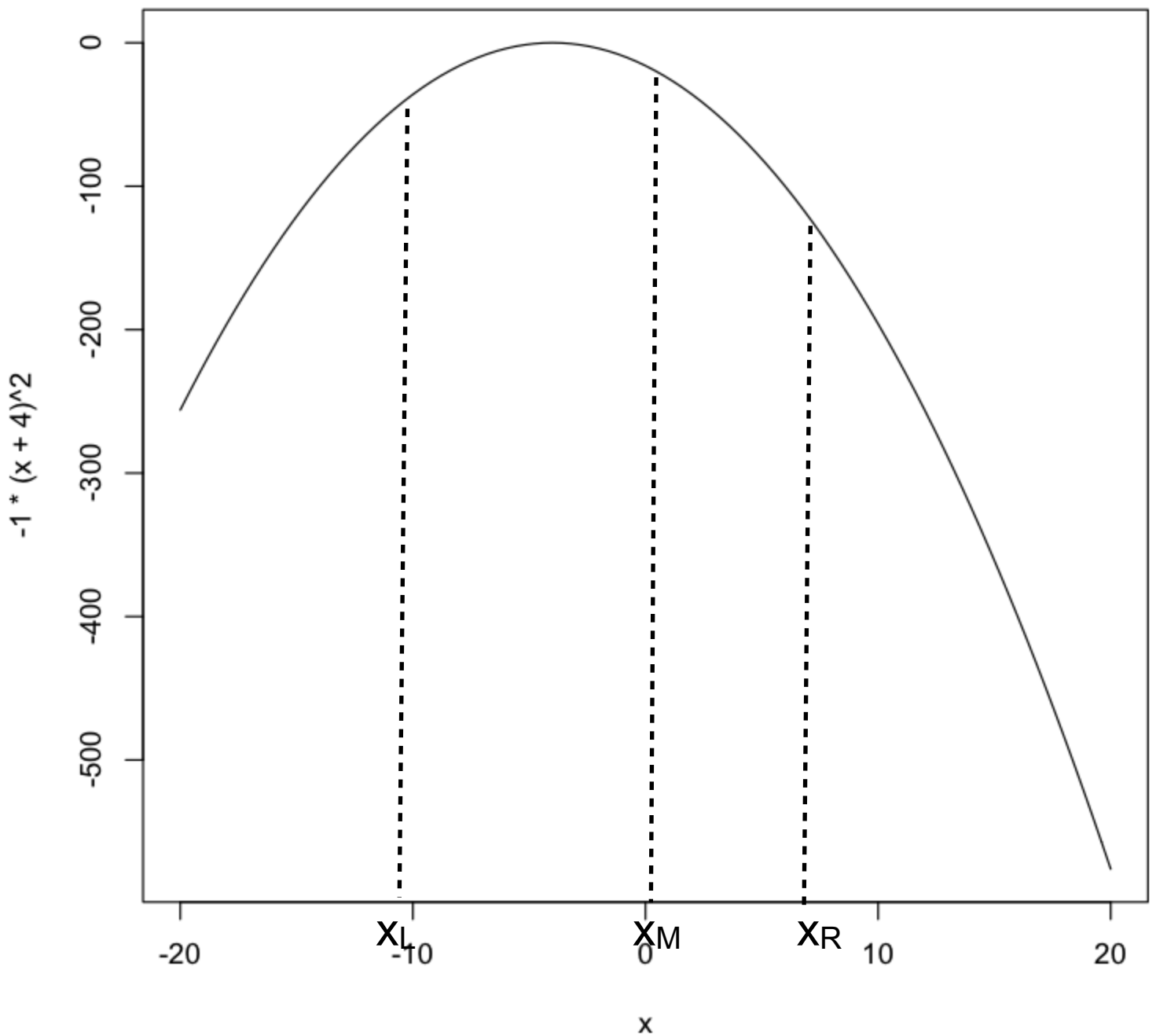


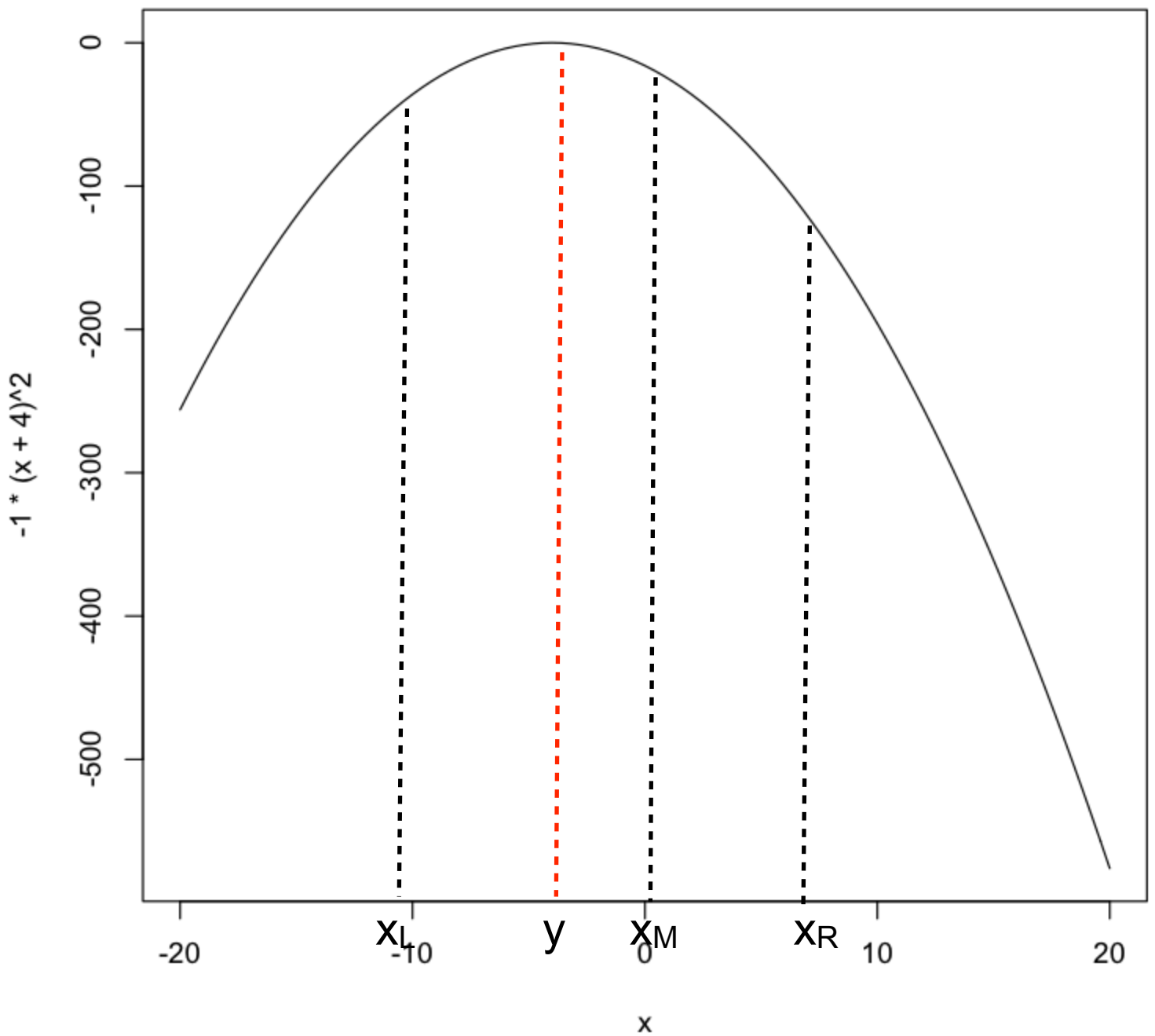
20

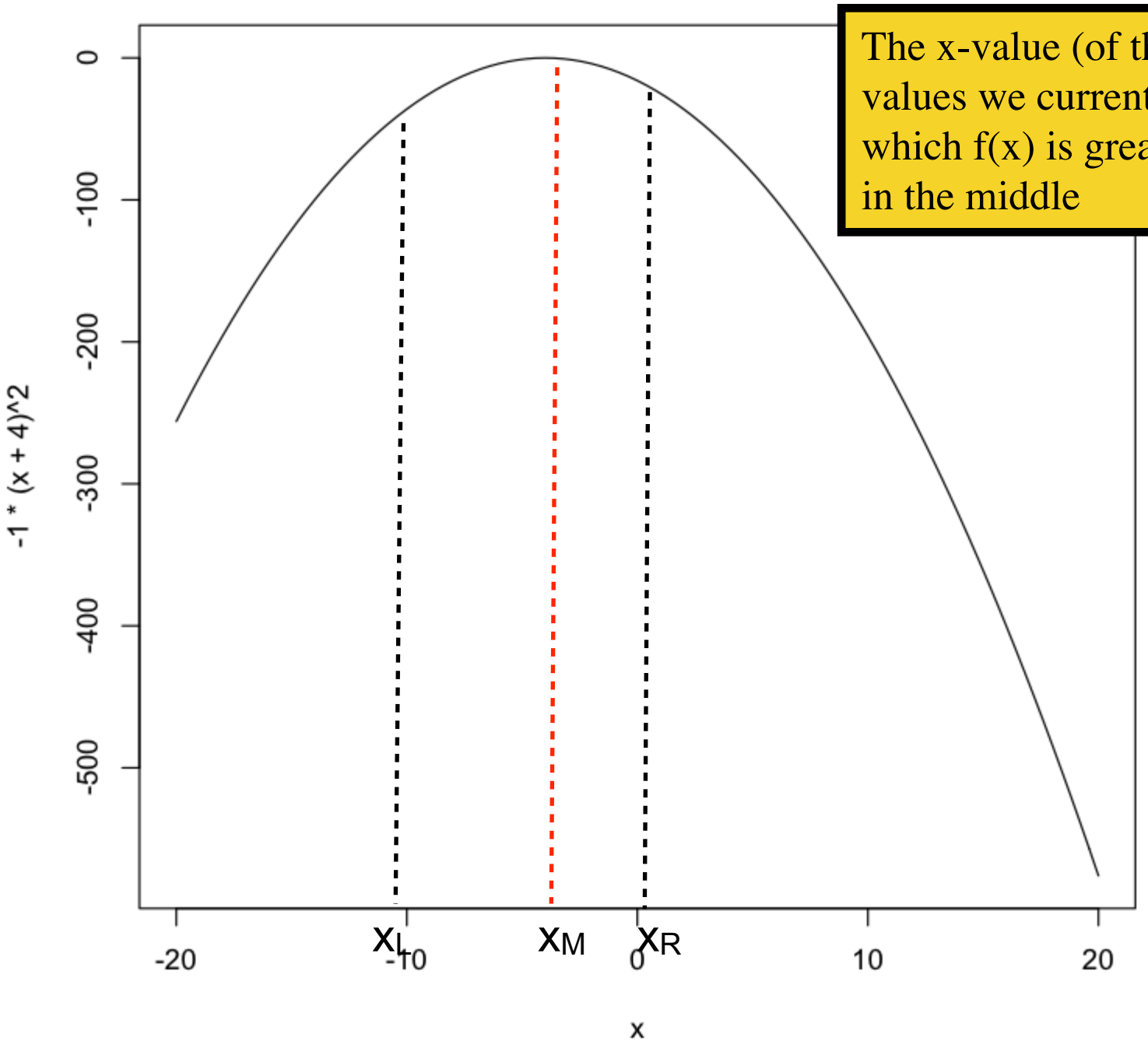

-1 * (x + 4)^2
0
-100
-200
-300
-400
-500
-20
-10
0
10
20
x
x_L
y
x_M
x_{R1}


-1 * (x + 4)^2
0
-100
-200
-300
-400
-500
-20
-10
0
10
20
x
XL
XM
XR


-1 * (x + 4)^2
0
-100
-200
-300
-400
-500
-20
-10
0
10
20
x
x_L
x_M
y
x_R

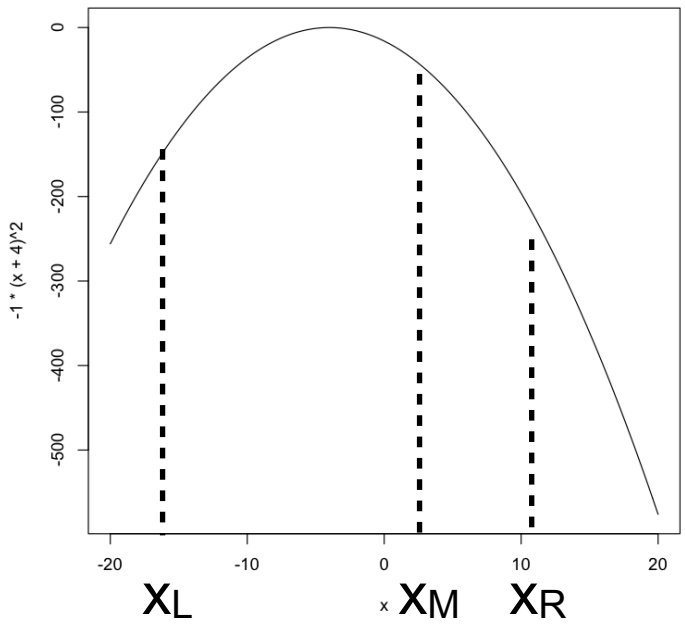

0
-100
-200
-300
-400
-500
-1 * (x + 4)^2
-20
-10
0
10
20
x
x_L
x_M
x_R


-1 * (x + 4)^2
0
-100
-200
-300
-400
-500
-20
-10
0
10
20
x
x_L
y
x_M
x_R



The x-value (of the three x-values we currently have) for which f(x) is greatest, is always in the middle

Golden section method



Start with x_L, x_M, x_R such that

$f(x_L) \le f(x_M) \ge f(x_R)$

1. If $x_R - x_L \le \varepsilon$ stop.
2. If $x_R - x_M > x_M - x_L$ then do 2a;

   otherwise do 2b.

   2a. Choose a $y \in (x_M, x_R)$.

   If $f(y) \ge f(x_M)$ then put $x_L = x_M$ and $x_M = y$; otherwise put $x_R = y$.

   2b. Choose a $y \in (x_L, x_M)$.

   If $f(y) \ge f(x_M)$ then put $x_R = x_M$ and $x_M = y$; otherwise put $x_L = y$.

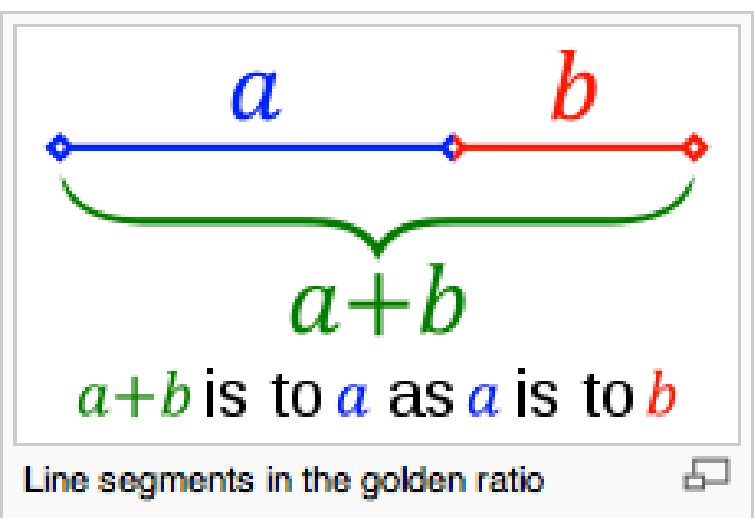
3. Go back to step 1.

27

How to choose y?

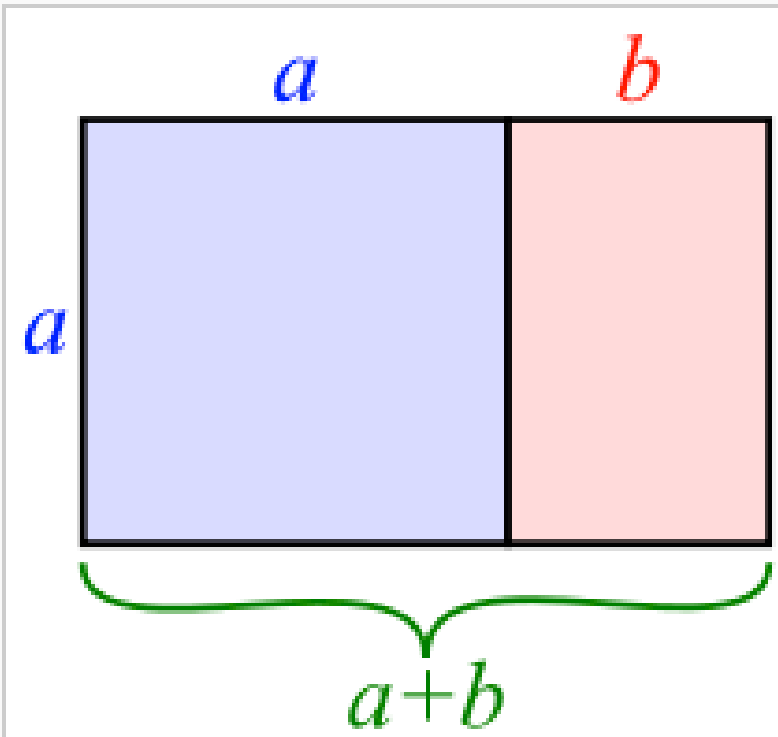
- Mid-point (Bisection method)

- Golden Ratio
  - Two quantities are in the golden ratio if the ratio of the sum of the quantities to the larger quantity is equal to the ratio of the larger quantity to the smaller one (wikipedia).



Line segments in the golden ratio

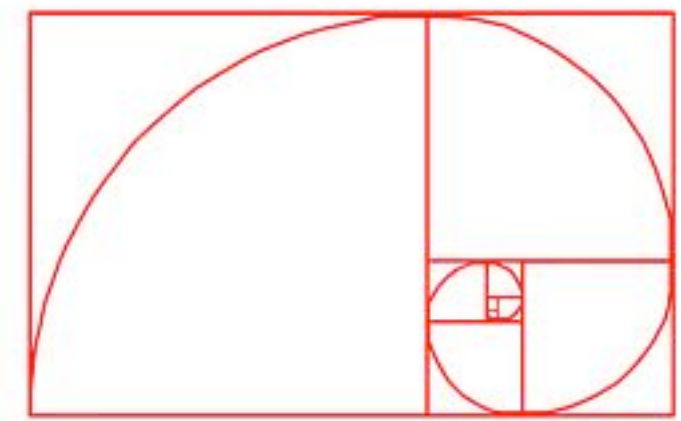
$$a/(a+b) = b/a$$



A golden rectangle with longer side *a* and shorter side *b*, when placed adjacent to a square with sides of length *a*, will produce a similar golden rectangle with longer side *a* + *b* and shorter side *a*. This illustrates the relationship $\frac{a+b}{a} = \frac{a}{b} \equiv \varphi$.

(Wikipedia)

Alternatively, if you start with a golden rectangle and then remove a square from one end, you have another (smaller) golden rectangle (ad infinitum).

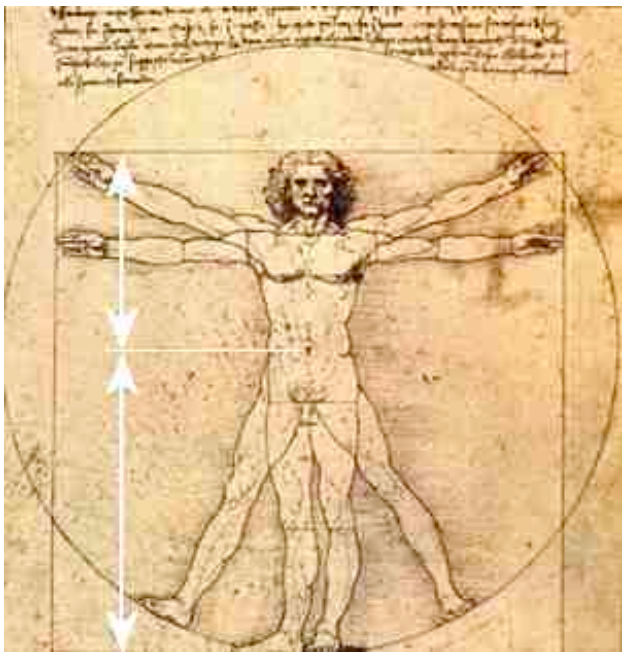


Golden spiral

How to choose y?

- Mid-point (Bisection method)

- Golden Ratio



How to choose y?

- Golden Ratio
  - Two quantities are in the golden ratio if the ratio of the sum of the quantities to the larger quantity is equal to the ratio of the larger quantity to the smaller one.
  - So, we choose a y such that the ratio of the width of the largest and smallest interval stays the same (whichever interval we end up removing!).
  - This means we choose according to the Golden ratio $(1+\sqrt{5})/2$.
  - (Worst-case) convergence is faster than for other choices.

Golden section method

- Start with x_L, x_M, x_R such that $f(x_L) \leq f(x_M) \geq f(x_R)$

1. If $x_R - x_L \leq \varepsilon$ stop.
2. If $x_R - x_M > x_M - x_L$ then do 2a; otherwise do 2b.

   2a. Choose a $y \in (x_M, x_R)$. $y = x_M + (x_R - x_M)/(1+\rho)$, where $\rho = (1+\sqrt{5})/2$.

   If $f(y) \geq f(x_M)$ then put $x_L = x_M$ and $x_M = y$; otherwise put $x_R = y$.

   2b. Choose a $y \in (x_L, x_M)$. $y = x_M - (x_M - x_L)/(1+\rho)$, where $\rho = (1+\sqrt{5})/2$.

   If $f(y) \geq f(x_M)$ then put $x_R = x_M$ and $x_M = y$; otherwise put $x_L = y$.

3. Go back to step 1.

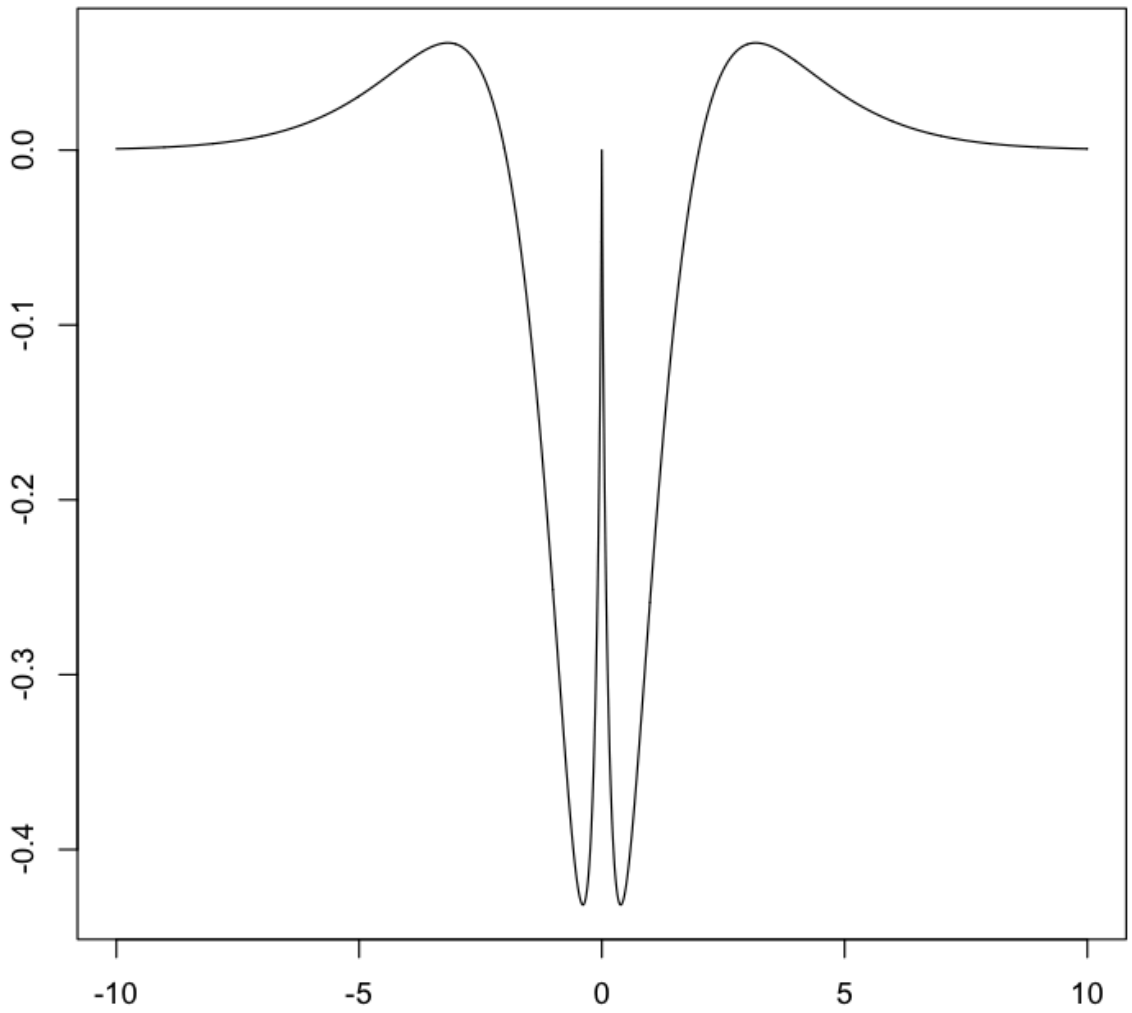
Golden Section Lab Task

- Code up the Golden Section optimization method.
- Test it on:

$f(x) = -(x+4)^2$ Find max

$$f(x) = \begin{cases} 0, & x = 0 \\ |x|\log(|x|/2)e^{-|x|}, & \text{otherwise} \end{cases}$$

Find max and min

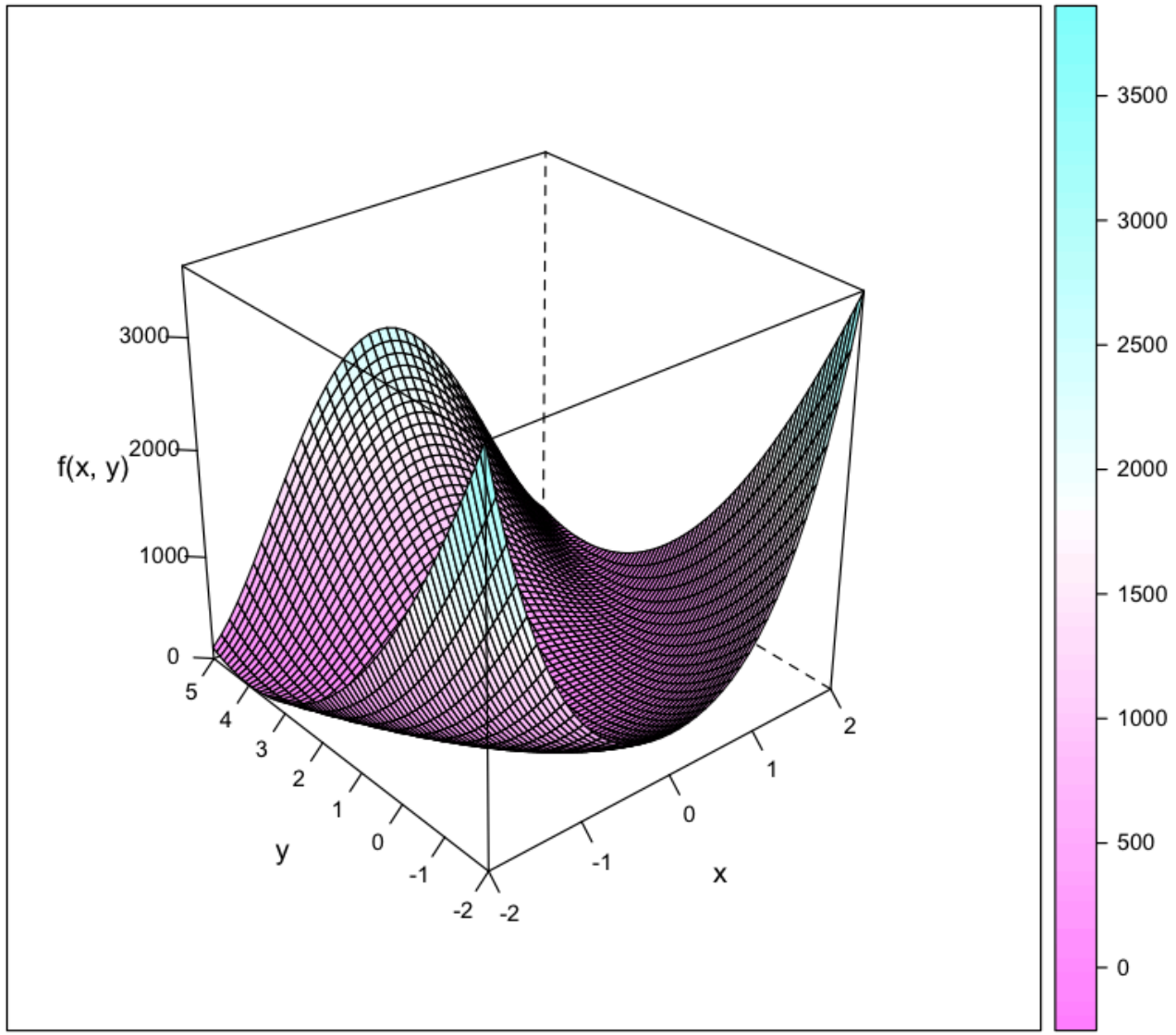


Find both max and min

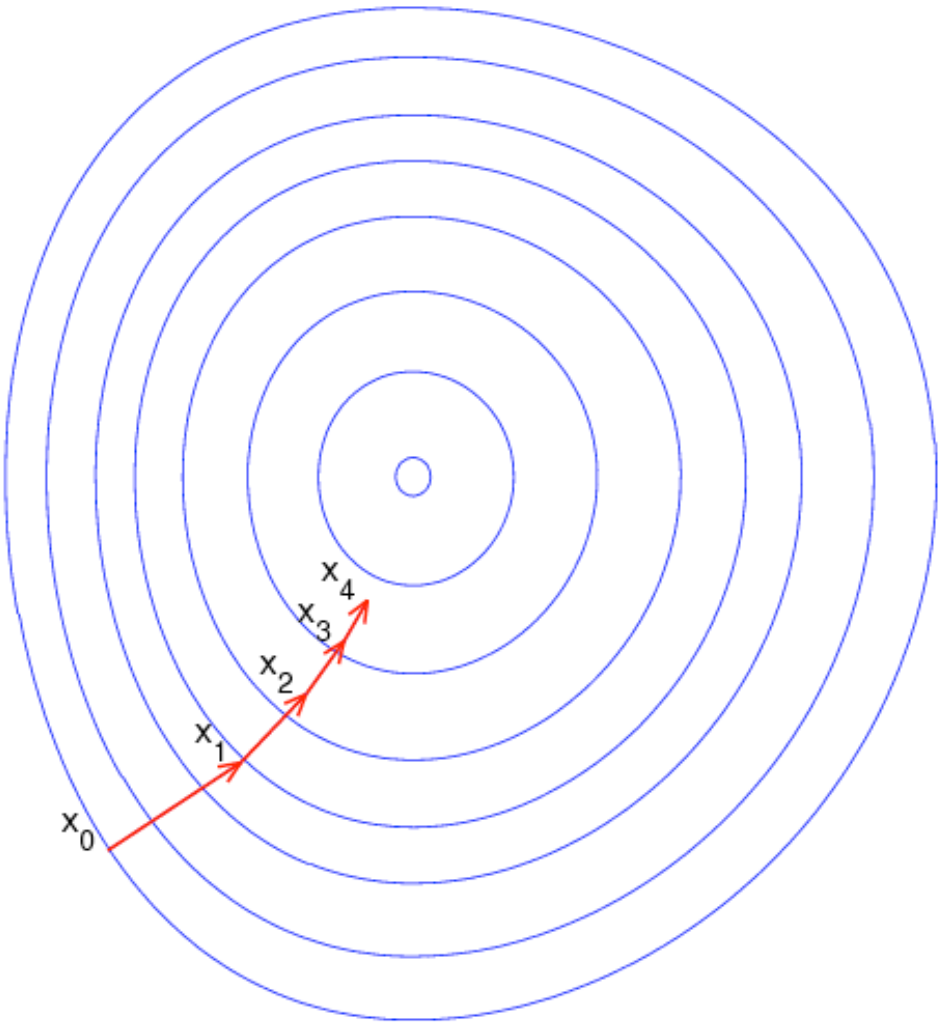
$$f(x) = \begin{cases} 0, & x = 0 \\ |x|\log(|x|/2)e^{-|x|}, & \text{otherwise} \end{cases}$$

25 minutes of Lab Time…

Methods for 2 dimensions

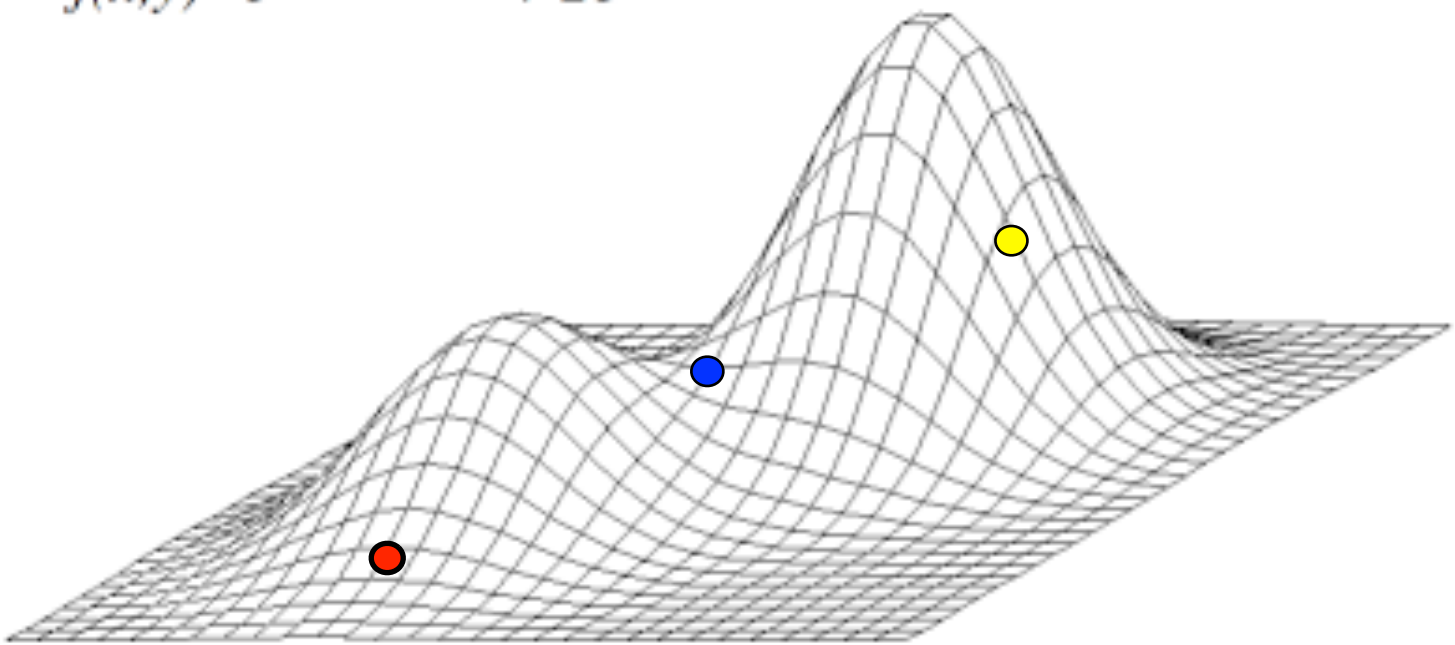


Steepest accent (hill-climbing)



Problem: local maximum

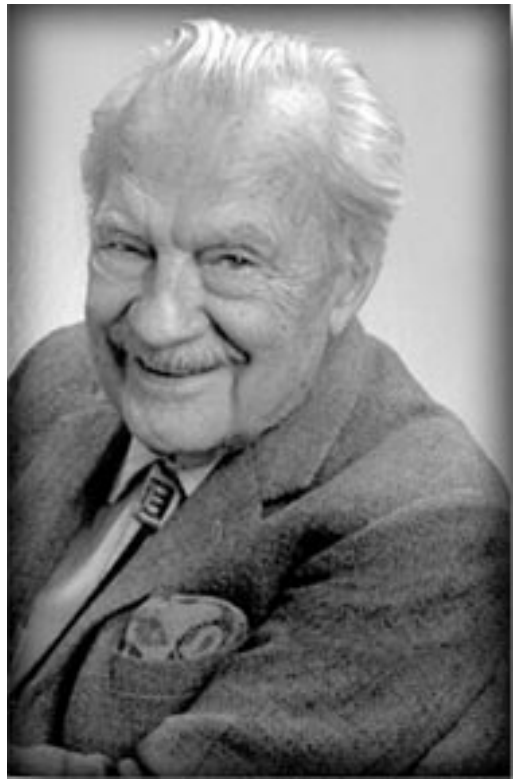
$$f(x,y)=e^{-(x^2+y^2)} + 2e^{-((x-1.7)^2+(y-1.7)^2)}$$



‘Solutions’:

- **Random-start hill-climbing**
- Iterate the following:
  1. Choose a random start point
  2. Run a hill-climbing algorithm
  3. If solution beats ‘best so far’, store it.

Metropolis algorithm (Metropolis, 1953)



Born: June 11, 1915
Chicago

- Define an 'energy' for a configuration of a system. Assume we are **minimizing** energy (denoted by f(x)).
- Propose moves from an old state x, to a new state x' according to a **symmetric** proposal kernel q(). [So q(x->x')=q(x'->x) for all x, x'].
- Let ΔE denote the change in energy between the new and old state. i.e., we define ΔE(x,x')=f(x')-f(x).
- Move from x to x' with prob. min{1,exp(-ΔE(x,x')/T)}, where T is a 'temperature' [FIXED]. *(This is for a minimizing algorithm, if you are maximizing use h=min{1,exp(ΔE(x,x')/T)}.)*
-

If you are minimizing:
when energy decreases, you move with prob. 1
when energy increases, you move with prob. <1.

Simulated annealing

- Kirkpatrick, S.; Gelatt, C. D.; and Vecchi, M. P. "Optimization by Simulated Annealing." Science 220, 671-680, 1983.
- Similar in spirit to the Metropolis algorithm.
- Again, uses a rule q for proposing moves to a new state x' based on the current state x. Assume we are trying to find the **minimum** of f.
- Define $\Delta E(x,x')=f(x')-f(x)$.
- Move to new state with prob. $h=\min\{1,\exp(-\Delta E(x,x')/T)\}$, where T is a 'temperature'. *(This is for a minimizing algorithm, if you are maximizing use $h=\min\{1,\exp(\Delta E(x,x')/T)\}$)*
- e.g. T=1:
  - $f(x')-f(x)=1 \rightarrow h=1/e < 1$
  - $f(x')-f(x)=-1 \rightarrow h=e > 1$
- **T is always positive, but decreases over time** (What starting value? At what rate should it decrease?) so moves in the wrong direction become less likely.

Simulated annealing - pseudocode

```
# Here's a function to practice minimizing. The minimum is at (0,0)
XYSquared<-function(x)
{
 return ((x[1]^2+x[2]^2))
}

# pass this function the name of the function you are trying to minimize ("Fn"), the starting coordinates
# ("StartPoint"), the initial temperature, funal temperature, and proportion by which to decrease the temperature
# each iteration
SimAnneal2D<-function(Fn,StartPoint,InitialTemp,FinalTemp,TempDecreaseRate){
 StepSize<-0.5   # how much are we going to move by for each step?
 X<-StartPoint[1]
 Y<-StartPoint[2]
 # record the value the function takes at this point (FnVal, say)
 # ADD code to calculate FnVal here

 # Set the temperature (Temp) to = InitialTemp,
 # ADD Code to set TEMP here

 plot(X,Y,xlim=c(-2*X,2*X),ylim=c(-2*Y,2*Y)) # set up a plot
 points(0,0,pch=19,col="red")  # plot the Minimum - this is where you are hoping to end up
 while (Temp>FinalTemp){  # we keep going until things cool down
  #propose new point
  NewX<-X+rnorm(1,mean=0,sd=StepSize)  # you don't have to do it this way, but it seems reasonable to me
  NewY<-Y+rnorm(1,mean=0,sd=StepSize)
  New<-c(NewX,NewY)
  #decide whether to move
  NewVal<- Fn(New)  # the value the function takes at the new point

  # Calculate the value of H
  # Add CODE FOR h HERE

     p<-runif(1)  # the random number that is going to help us decide whether to move
  if (p<h){ # we move
    # add an arrow to the plot showing where we moved:
   arrows(X,Y,NewX,NewY,length=0.05)
   # update your records of where we are:...X<-NewX   Y<-NewY
   # Add code here

   # pause for a bit - otherwise the plots flash by too quickly to see properly
   Start.Time<-Sys.time()
   while (Sys.time()<Start.Time+0.02){}   # there is probably a better way of doing this
   # plot the path you are taking
   points(X,Y,pch=19,col="blue")

   # update the record of the function value:
   FnVal<-NewVal
  }
  #reduce temperature
  Temp<-Temp*(1-TempDecreaseRate)

 }
 points(X,Y,pch=19,col="blue")   # a big blue point to show where we ended up
 return (c(X,Y,Fn(X,Y)))  # return the coordinates of our final resting place, and the function value
}

# Call the function like this (for this example we start at the point (5,5), with a temperature going from 10 to 0.1)
SimAnneal2D(XYSquared,c(5,5),10,0.1,0.02)
```

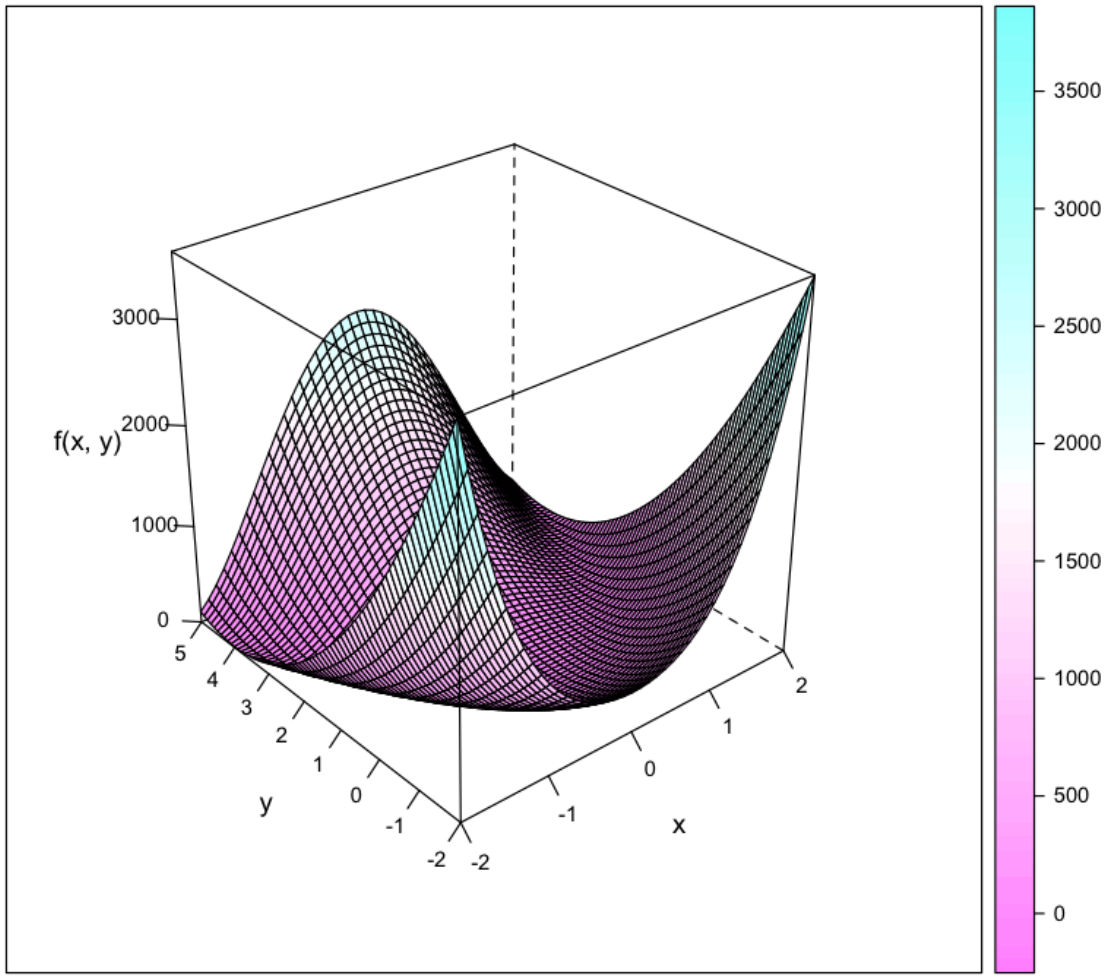
SimAnnealPseudoCode.Rmd on Github

SA or Metropolis lab task #1

- Code up either the SA or Metropolis algorithm. Test it on a simple function, such as $f(x,y)= x^2+y^2$
- Then test it on the Rosenbrock function (see next slide)

Rosenbrock function

$$f(x,y)=(1-x)^2+100(y-x^2)^2$$



Commonly-used test function.

Has a global minimum at (1,1).

```
source('Rosenbrock.r')
MyRosenbrock<-function(x){
   return (Rosenbrock(x)[[1]])
}
```

SA or Metropolis task #2: regression

Regression fit aims to minimize sum of squared residuals:

$\mathbf{x}=\{x_1,x_2,x_3,\ldots,x_n\}$

$\mathbf{y}=\{y_1,y_2,y_3,\ldots,y_n\}$

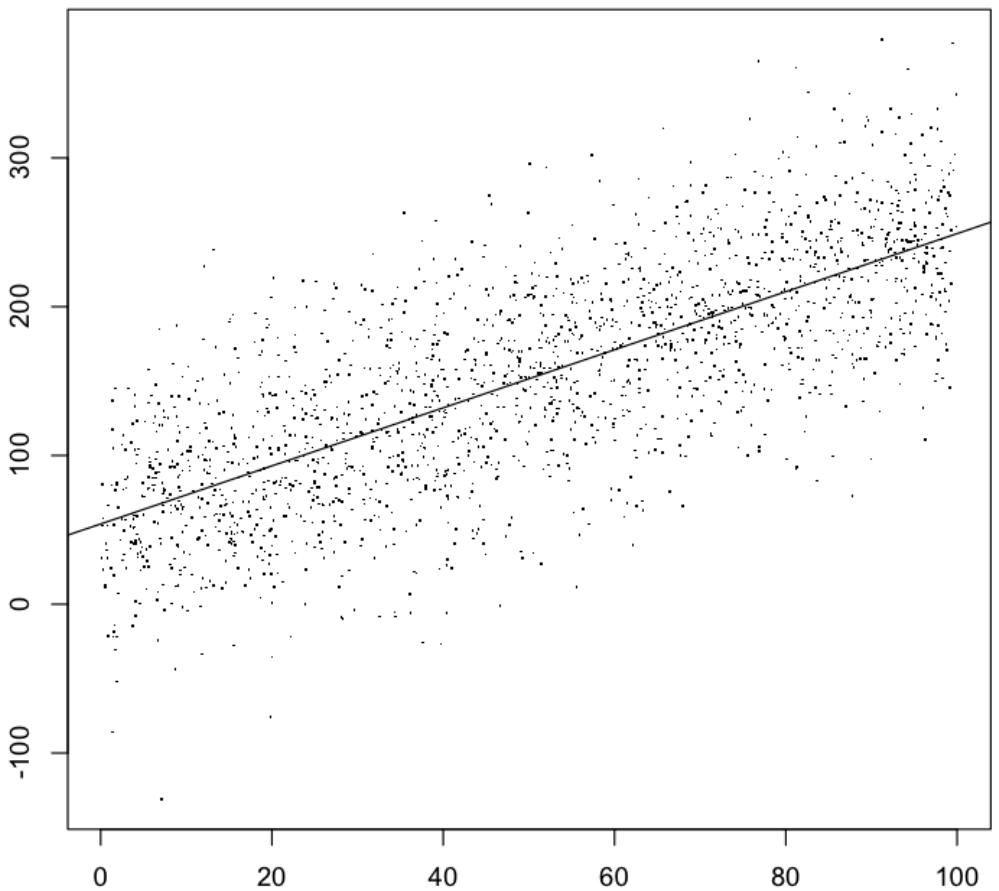
a=intercept; b=slope; so y=ax+b

$SS(a,b)=\sum_i$ (Observed y_i - Fitted y)^2

So, search through space of values for (a,b) (i.e., Intercept, slope), to find minimal (a,b) that give minimal SS(a,b).

Work in small groups if you wish to.

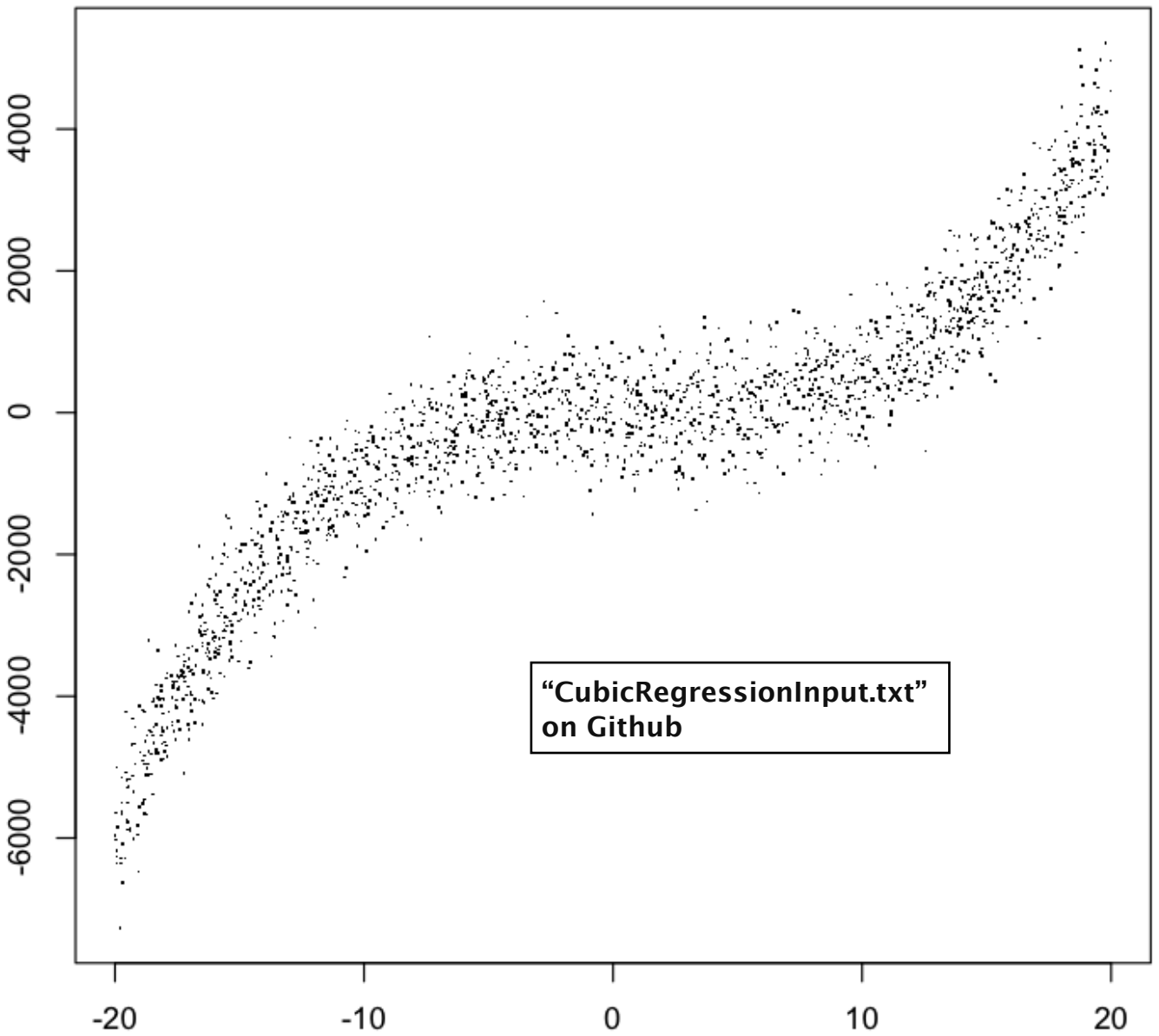
Linear Regression



On github: “RegressionInput.txt ”

```
filestr<-sprintf("RegressionInput.txt")
#filestr<-sprintf("CubicRegressionInput.txt")
Coordinates<-read.table(filestr,header=F)
```

Cubic Regression



4 terms:
Intercept
Linear coefft.
Quadratic coefft.
Cubic coefft.

Now have to optimize in 4 dimensions (this will be harder).

30(?) minutes of Lab Time…

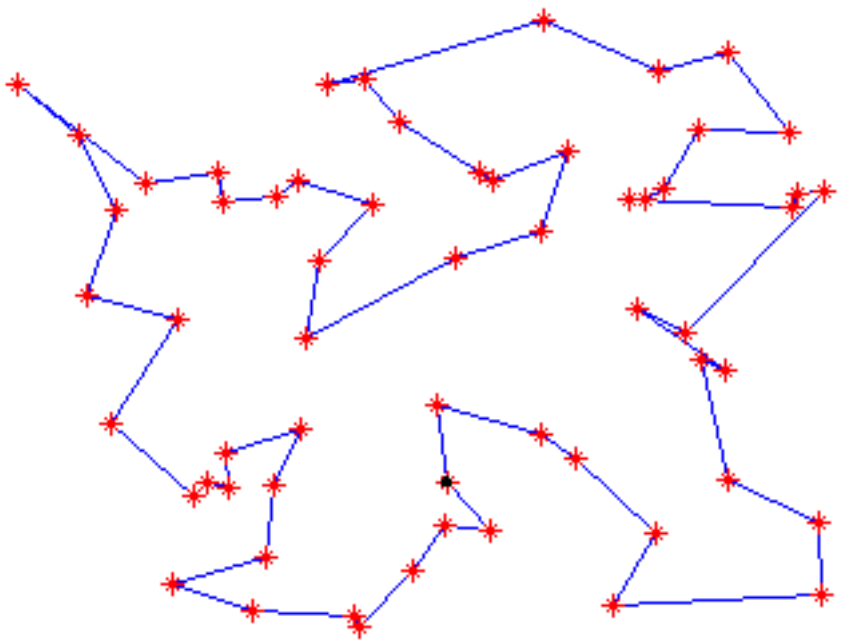
Potential end of term presentation/report topic - 4 color problem

- The 4-color problem: open conjecture that any map of states (e.g.) can be colored, such that adjacent states never have the same color, using at most 4 colors.

Potential end of term presentation/ report topic - Traveling salesperson problem

- Given a set of cities, with known distances between them, what is the shortest round-trip route that visits all cities?

Traveling salesperson problem



- Historical note: www.math.uwaterloo.ca/tsp/gallery/igraphics/car54.html

Traveling Salesperson

- Write an optimization routine that attempts to solve the Traveling Salesman problem

- Test data is uploaded to Blackboard ('TSPdata10towns.txt', 'TSPdata20towns.txt', 'TSPdata50towns.txt', 'TSPdata100towns.txt') [Note: This problem is "NP hard"]

- Need to define an 'energy' function. (The length of the path. It is this you are trying to minimize.)
- Need to define a rule q(x -> x'), where x,x' are paths through the towns. This is how you will change your path from iteration to iteration of the optimization algorithm.

- Fame at last: http://www.send2press.com/newswire/2007-07-0709-002.shtml
-

Sudoku

	7		1		5			
5						3		
		6	7	9		1	2	5
1	5	2	8		6		9	
				3			5	
3				5			8	
8	6			7		5	1	
				6				
		9			2		4	

Sudoku

9	7	3	1	2	5	4	6	8
5	2	1	6	8	4	3	7	9
4	8	6	7	9	3	1	2	5
1	5	2	8	4	6	7	9	3
6	4	8	9	3	7	2	5	1
3	9	7	2	5	1	6	8	4
8	6	4	3	7	9	5	1	2
2	1	5	4	6	8	9	3	7
7	3	9	5	1	2	8	4	6

Sudoku rules

9	7	3	1	2	5	4	6	8
5	2	1	6	8	4	3	7	9
4	8	6	7	9	3	1	2	5
1	5	2	8	4	6	7	9	3
6	4	8	9	3	7	2	5	1
3	9	7	2	5	1	6	8	4
8	6	4	3	7	9	5	1	2
2	1	5	4	6	8	9	3	7
7	3	9	5	1	2	8	4	6

- Each 3x3 square must contain all the numbers 1,2,3,4,5,6,7,8,9.

Sudoku rules

9	7	3	1	2	5	4	6	8
5	2	1	6	8	4	3	7	9
4	8	6	7	9	3	1	2	5
1	5	2	8	4	6	7	9	3
6	4	8	9	3	7	2	5	1
3	9	7	2	5	1	6	8	4
8	6	4	3	7	9	5	1	2
2	1	5	4	6	8	9	3	7
7	3	9	5	1	2	8	4	6

- Each 3x3 square must contain all the numbers 1,2,3,4,5,6,7,8,9.

Sudoku rules

9	7	3	1	2	5	4	6	8
5	2	1	6	8	4	3	7	9
4	8	6	7	9	3	1	2	5
1	5	2	8	4	6	7	9	3
6	4	8	9	3	7	2	5	1
3	9	7	2	5	1	6	8	4
8	6	4	3	7	9	5	1	2
2	1	5	4	6	8	9	3	7
7	3	9	5	1	2	8	4	6

- Each row must contain all the numbers 1,2,3,4,5,6,7,8,9.

Sudoku rules

9	7	3	1	2	5	4	6	8
5	2	1	6	8	4	3	7	9
4	8	6	7	9	3	1	2	5
1	5	2	8	4	6	7	9	3
6	4	8	9	3	7	2	5	1
3	9	7	2	5	1	6	8	4
8	6	4	3	7	9	5	1	2
2	1	5	4	6	8	9	3	7
7	3	9	5	1	2	8	4	6

- Each row must contain all the numbers 1,2,3,4,5,6,7,8,9.

Sudoku rules

9	7	3	1	2	5	4	6	8
5	2	1	6	8	4	3	7	9
4	8	6	7	9	3	1	2	5
1	5	2	8	4	6	7	9	3
6	4	8	9	3	7	2	5	1
3	9	7	2	5	1	6	8	4
8	6	4	3	7	9	5	1	2
2	1	5	4	6	8	9	3	7
7	3	9	5	1	2	8	4	6

- Each column must contain all the numbers 1,2,3,4,5,6,7,8,9.

Sudoku rules

9	7	3	1	2	5	4	6	8
5	2	1	6	8	4	3	7	9
4	8	6	7	9	3	1	2	5
1	5	2	8	4	6	7	9	3
6	4	8	9	3	7	2	5	1
3	9	7	2	5	1	6	8	4
8	6	4	3	7	9	5	1	2
2	1	5	4	6	8	9	3	7
7	3	9	5	1	2	8	4	6

See
"TestSudoku.txt"
"SudokuPlot.R"
on Blackboard

- Each column must contain all the numbers 1,2,3,4,5,6,7,8,9.

- How could you attack this problem as an optimization problem?

Sudoku

- Test data is uploaded to Github (“TestSudoku.txt”), along with a function to draw your sudoku to the screen (“SudokuPlot.R”)
- Need to define an ‘energy’ function. (How good a given solution is.) What might you use for this?
- Need to define a rule q(x -> x’), where x,x’ are potential solutions. This is how you will change your solution from iteration to iteration of the optimization algorithm. This is the bit that will probably determine how well your algorithm works.
- Need to choose a configuration from which to start the algorithm.

Lab Task 3 + potential end of term presentation) - Sudoku challenge

- Write a function that uses optimization to solve Sudoku.
- Virtually complete “pseudo”-code is on Github “SudokuPseudoCode.R”. You just need to write a function to change the configuration of numbers in the grid.
- Test it on the Sudoku in “Sudoku_simple.txt”

- Can you solve the ‘moderate’ (or hard or diabolical) sudoku that is also uploaded?

Potential end of term presentation - The Grasshopper problem

A grasshopper lands somewhere randomly on your lawn, which has an area of 1 square meter.

As soon as it lands, it jumps 30 centimeters.

What shape should your lawn be to maximize the chances that the grasshopper will still be on the lawn after the 30-centimeter jump?

(Hint: It's not a circle.)

Extra credit: What if the grasshopper jumps X centimeters instead?

How do we attack this problem as an optimization problem?

Remainder of class is lab time

END