Algorithms and Data Structures

Charles A. Wuethrich

Bauhaus-University Weimar - CogVis/MMC

June 20, 2019

Mathematical Algorithms

- Introduction
- Matrix based
  - Transitive hull
  - All shortest paths
  - Gaussian elimination
- Random numbers
- Interpolation and Approximation
  - Spline interpolation
  - Minimum squares
  - Integration

Introduction

- There is an additional category of algorithms that bases its methods on mathematics
- These algorithms differ from the previous in that they use mathematics to solve a problem
- Some of them use vector and are statistics based
- Some of them transform matrices
- Some other use approximation to get a solution which is as close as possible to the exact solution

Matrix based

Transitive hull

- Let us go back to graphs
- Remember most of the algorithms based on stacks or queues?
- Well, the next problem we introduce uses instead matrix operators for the solution of a problem
- The problem is called the problem of the transitive hull:
    - which nodes can be reached from one node?

Transitive hull

- Suppose we have a directed graph
- Problem: which vertices can be reached from a given node?
- The visiting method used for depth search in normal graphs can be used also in directed graphs.
- Speed: $O(V(E+V))$ for non dense graphs $O(V^3)$ for dense graphs

- Simplest program to compute trans. hull

```
FOR y:=1 TO v DO
   FOR x:=1 TO v DO
      IF a[x,y] THEN
         FOR j:=1 TO v DO
            IF a[y,j] THEN a[x,j]:= TRUE
```

Idea: if there is a path $x \to y$, and one $y \to j$, then there is a path $x \to j$.
In the program$x \to y$ is done on smaller indices of y, and what it finds is paths using only nodes of indices smaller of $y + 1$

Transitive hull: Warshall Algo

- Walshall's algo. transforms the adjacency matrix of a graph into the adj. matrix of its transitive hull
- Start from start node
- Each row having a 1 at column y is replaced by the result of OR-ing it with the y-th row
- The resulting matrix tells which pairs are mutually reachable
- Speed: $O(V^3)$ for dense graphs

All shortest paths

- For weighted graphs it may be of interest to have a table listing the shortest path between all pair of nodes (all shortest paths prob.)
- Use shortest path algorithm for each vertices. Resulting complexity: $O((E+V)VlogV)$

- One can do better, by using a similar algo to Walshall's algo. This is called Floyd's algo.

```
FOR y := 1 TO v DO
   FOR x := 1 to v DO
      IF a[x,y] > 0 THEN
         FOR j=1 TO v DO
            IF a[y,j]>0 THEN
               IF ((a[x,j]=0) OR (a[x,y]+a[y,j]<a[x,j])
                  THEN a[x,j]:=a[x,y]+a[y,j];
```

Speed: $O(V^3)$

A small remark on Matrix Computations

- To compute matrix multiplication one has to remember which components give which resulting component
- Thus, if $R = P * Q$, the element $r[i,j]$ is the scalar product of the i-th row of p and of the j-th column of q
- Since for each of the N^2 elements of the resulting matrix N mults are necesssary, complexity is N^3
- Newer algorithms split the $N * N$ Matrix mult. into the sums of 4 quarter-matrices, so as to decrease complexity to $N^{lg7} \sim N^{2,81}$

Matrix inversion

- Suppose the following system is given:
  $a_{11}x_1 + a_{12}x_2 + \ldots + a_{1N}x_N = b_1$
  $a_{21}x_1 + a_{22}x_2 + \ldots + a_{2N}x_N = b_2$
  $\vdots$
  $a_{N1}x_1 + a_{N2}x_2 + \ldots + a_{NN}x_N = b_N$
- Or, in matrix form:

$$\begin{bmatrix} a_{11} & a_{12} & \ldots & a_{1N} \\ a_{21} & a_{22} & \ldots & a_{2N} \\ \vdots & & & \\ a_{11} & a_{12} & \ldots & a_{1N} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_N \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_N \end{bmatrix}$$

$$A\underline{x} = \underline{b}$$

- Problem: a solution of the system has to be computed, I.e.

$$\underline{x} = A^{-1}\underline{b}$$

- Note that the sol can be one, none or infinite
- As we know from math, computing the inverse is not trivial

Gaussian elimination

- Idea: make variables disappear from most equations
- Let us make an example:
$$\left\{\begin{array}{c} x_1 + 3x_2 - 4x_3 = 8 \\ x_1 + x_2 - 2x_3 = 2 \\ -x_1 - 2x_2 + 5x_3 = -1 \end{array}\right\}$$
- subtract 2nd line from first
$$\left\{\begin{array}{c} x_1 + 3x_2 - 4x_3 = 8 \\ 2x_2 - 2x_3 = 6 \\ -x_1 - 2x_2 + 5x_3 = -1 \end{array}\right\}$$
- add 3rd line to first
$$\left\{\begin{array}{c} x_1 + 3x_2 - 4x_3 = 8 \\ 2x_2 - 2x_3 = 6 \\ x_2 + x_3 = 7 \end{array}\right\}$$
- finally subtract double of 3rd line from second
$$\left\{\begin{array}{c} x_1 + 3x_2 - 4x_3 = 8 \\ 2x_2 - 2x_3 = 6 \\ -4x_3 = -8 \end{array}\right\}$$
- Now we can solve last one, with it the 2nd and with it 1st

Gaussian elimination

- Let us redo this in matrix form

$$\begin{bmatrix} 1 & 3 & -4 \\ 1 & 1 & -2 \\ -1 & -2 & 5 \end{bmatrix} \underline{x} = \begin{bmatrix} 8 \\ 2 \\ -1 \end{bmatrix}$$

- subtract 2nd line from first

$$\begin{bmatrix} 1 & 3 & -4 \\ 0 & 2 & -2 \\ -1 & -2 & 5 \end{bmatrix} \underline{x} = \begin{bmatrix} 8 \\ 6 \\ -1 \end{bmatrix}$$

- add 3rd line to first

$$\begin{bmatrix} 1 & 3 & -4 \\ 0 & 2 & -2 \\ 0 & 1 & 1 \end{bmatrix} \underline{x} = \begin{bmatrix} 8 \\ 6 \\ 7 \end{bmatrix}$$

- finally subtract double of 3rd line from second

$$\begin{bmatrix} 1 & 3 & -4 \\ 0 & 2 & -2 \\ 0 & 0 & -4 \end{bmatrix} \underline{x} = \begin{bmatrix} 8 \\ 6 \\ -8 \end{bmatrix}$$

- Once I have a triangular matrix, I can substitute my way upwards

Gaussian elimination

- What did we do?
  - forward elimination, to obtain a triangular matrix
  - backwards substitution
- How does forward elimination work?
  - First I eliminated first variable from all eq. except the first
  - Then eliminated 2^{nd} var. from all eq except first 2
  - and so on. . . ..
- To eliminate the i-th variable from the j-th equation, one
  - multiplies i-th row by a_{ji}/a_{ii}
  - subtract i-th row from j-th
- NB: a_{ii} should be $\neq 0$, if it is 0, swap line with another one that has $a_{ii} \neq 0$
- If none is found, the matrix is singular, and the system has solutions
- a_{ii} is called pivot

Gaussian elimination

- What did we do?
  - In fact, due to numerical errors in computations, it is best to choose among the remaining untouched lines the one with the biggest a_{ii}
  - Once the matrix is triangularized,one has to do backward substitution
- Given a system of N equations in N variables, the algorithm does ca.$N^3/3$ multiplications and additions
- Substitution runs in $O(N^2)$

Random numbers

Random numbers

- Note: do not confuse an arbitrary with a random number
- Arbitrary: not important which one it is
- Random: every number has the same probability $\Rightarrow$ The interval must be finite
- Normally, a sequence of random numbers is required (random sequence)

Random numbers

- One can program on a computer “similar” sequences to random sequences (pseudo-random numbers)
- There is a second category of “random” numbers: quasi-random numbers: do not have all the characteristics of random numbers, but only “interesting ones”

Random numbers

- What does random in reality mean? NOT that if we extract out of 100 numbers, each one will be drawn in 100 draws
- Instead, that we have equally distributed numbers, i.e. each number is equally probable
- Many test to prove this, we will use the χ^2-test

- In criptology for example the more a message looks random, the better
- Simulating program behaviour with random input data (random test)
- Simulation of complex systems
- Lotto, lotteries in general

Linear congruence

- Due to Lehmer (51), also called Lehmer's rest class method
- Let s be an arbitrary number (seed)
- Let m be a constant (modulo), and b another constant
- Let $s_0 = s$, and $s_{i+1} = (s_i * b + 1) MODm$
- Fast on computers, since the MOD operation is easy and fast (bit overflow)

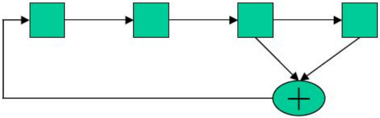
Linear congruence

- Difficult is the choice of the seed, and of m and b.
- Loads of studies on the subject
- m must be big
  - for example as big as a computer word
  - or a big power of 2 or of 10
- b must be neither too small nor too big
  - for example a number with 1 digit less than m
  - AND with non repetitive patterns in its digits
  - AND ending with x21, whereby x is even
- This to avoid cycles

- Take a look at D. Knuth's book: the Art of Computer Programming
- By the implementation watch be careful to overflows

Additive congruence

- Derived for old coding machines
- Start from a binary number, shift to right
- Insert new number by XOR with most right two digits



1111	1000	1100	1010
0111	0100	0110	1101
0011	0010	1011	1110
0001	1001	0101	1111

Additive congruence

- Note that all bit sequences appear
- Cyclic
- In general, for n bits word length, it is possible to set up things so that the cycle length is $2^n - 1$
- Lots of research was done to know which initial seed produces all bit patterns

Additive congruence

- Note that if we XOR two subsequent words of the sequence, we obtain the word that will appear three steps afterwards in the seq.
- This is easily implementable recursively

Randomness test

- To test the randomness of the pseudo-random generators the χ^2-test is used.
- The idea is the following:
  - Suppose N numbers are generated($N < r$).
  - Then per value we should get N/r numbers
  - BUT frequencies shouldn't be all the same, because this is not random.

- We compute the sum of the squares of the frequencies, divide it by expected frequencies, and subtract the size of the sequence.
- This is called χ^2-statistics

$$\chi^2 = \frac{\sum_{0 \le i \le r} (f_i - N/r)}{N/r}$$

If close to r, then sequence is random

Implementation Remarks

- Often two different pseudo-random generators are used.
- In this case, one generator achieves a sequence and the second one chooses positions in the sequence
- Random generators have all their problems

Interpolation and Approximation

- One common problem scientists have is to find a “well behaving” curve fitting certain data
- And sometimes it is not possible to compute all results precisely, but only an approximation of the solution is needed
- The quality of the approximation is how close it is to the exact solution

- The interpolation problem:
  - given N points (x_i, y_i) find the curve passing by them.
- Clearly, there is one and only one polynomial of degree N-1 that does this
- Classic solution: Lagrange interpolation

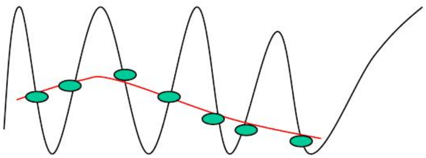
$$p(x) = \sum_{1 \leq i \leq N} y_i \prod_{1 \leq j \leq N, j=i} \frac{x - x_j}{x_i - x_j}$$

- This in fact looks more complicated then what it is
- Let us compute the polynomial passing through (1,3), (2,7), (3,13)

$$p(x) = 3\frac{x-2}{1-2}\frac{x-3}{1-3} + 7\frac{x-1}{2-1}\frac{x-3}{2-3} + 13\frac{x-1}{3-1}\frac{x-2}{3-2} = x^2 + x + 1$$

Interpolation

- What is interpolation good for?
- For example, in experim. setups: a certain function is measured at certain points, but values of this function at other points inbetween are required
- Problem with N-1 degree polynomials is that they wiggle too much (obscillate)
- Thus other methods of interpolation have been used, which use lower order curves



Spline interpolation

- The obscillation problems can be solved by using piecewise curves joining in a continuous way
- Use 3rd order curves $s_i(x) = a_i x^3 + b_i x^2 + c_i x + d_i$ i=1,2,...,N-1, where si interpolates between x_i and $x_i + 1$

Spline interpolation

- The splines must satisfy certain conditions:
  - They must pass through the points to be interpolated $s_i(x_i) = y_i$ and $s_i(x_{i+1}) = y_{i+1}$ (i=1,...N-1)
  - No abrupt changes in the derivative at the joints $s'_{i-1}(x_i) = s'_i(x_i)$ (i=2,...,N-1)
  - Also their 2nd derivatives must coincide $s_{i-1}''(x_i) = s_i''(x_i)$ (i=2,...,N-1)
  - This gives a system of 4N-6 equations in 4(N-1) unknown coefficients
  - Add two conditions (natural splines) $s_1''(x_0) = s_{N-1}''(x_N) = 0$

Spline interpolation

- Same can be computed more efficiently, since de facto there are only N-2 unknowns:
  - most conditions are redundant
  - For example, if p_i is the second order derivative at x_i $s''_{i-1}(x_i) = s''_i(x_i) = p_i$ (i=2,..,N-1), and $p_1 = p_N = 0$, then all a_i, b_i, c_i, d_i can be computed, since

$$s_i(x_i) = y_i$$
$$s_i(x_{i+1}) = y_{i+1}$$
$$s_i''(x_i) = p_i$$
$$s_i''(x_{i+1}) = p_{i+1}$$

Spline interpolation

- x- and y-values are known we need to know $p_2, .., p_{N-1}$
- to do this, we use the other condition on equal 1st derivatives at joins: this gives N-2 additional cond. to find out the N-2 values pi
- Now, if we wanted now to find the coefficients a_i, b_i, c_i, d_i from the p_i for every single spline, we would get pretty complicated expressions
- Instead of this, we can use a canonical form that includes few unknown coeffs.

Spline interpolation

- Let $t = (x - x_i)/(x_{i+1} - x_i)$
- With this substitution, the spline becomes

  $s_i(t) - ty_{i+1} + 1 + (1 - t)y_i + \frac{(x_{i+1} - x_i)^2}{6}((t^3 - t)p_{i+1} - ((1 - t)^3 - (1 - t))p_i$
- which means, that each spline is now defined in the interval [0,1] of t
- The eq looks terrible, BUT we are interested mostly in points where t=0 or t=1, where either t or (1-t) are 0
- Obviously, the resulting curve is continuous and its derivatives are continuous

Spline interpolation

- It also passes through the y_i, so we will use these functions
- Let us write the first derivative

$$s_i'(t) = z_i + (x_{i+1} - x_i)((3t^2 - 1)p_{i+1} - (3(1-t)^2 - 1)p_i)/6$$

- where $z_i = (y_{i+1} - y_i)/(x_{i+1} - x_i)$
- Now, if we set $s_{i-1}'(1) = s_i'(0)$ we get the system of N-2 eq

$$(x_i - x_{i-1})p_{i-1} + 2(x_{i+1} - x_{i-1})p_i + (x_{i+1} - x_i)p_{i+1} = 6(z - z_{i-1})$$

in the pi which can be solved in linear time (symmetric tridiagonal system)

Spline interpolation

- Note that here we use matrix solution for computing the spline coefficients
- Again, a method builds upon another one
- This will be done also for some of the next methods

Minimum squares

- Sometimes, the data (points) are not exact, but one needs a fitting curve to the data (x_i, y_i)
- In this case, it is convenient to use as functions family linear combinations of simple functions:$f(x) = c_1 f_1(x) + ... + c_M f_M(x)$
- The most common method is called the minimum squares method
- The idea is to minimize the vertical distances between fitting points and values of the function $f(x_i)$
- Basically, one has to find the minimum of the function of the coefficients c_i thus to find the place where the derivative is 0

- Let us suppose that we have to fit a function of the form $f(x) = c_1 f_1(x) + c_2 f_2(x)$ near the points $(x_1, y_1)(x_2, y_2)(x_3, y_3)$
- Task is to find the coefficients c_1, c_2 such that the square error

  $$E = (c_1 f_1(x_1) + c_2 f_2(x_1) - y_1)^2 + (c_1 f_1(x_2) + c_2 f_2(x_2) - y_2)^2 + (c_1 f_1(x_3) + c_2 f_2(x_3) - y_3)^2$$

  is minimum

- To do this, we need to set at the same time dE/dc_1 and dE/dc_2 to zero

$\frac{dE}{dc_1} = 2(c_1 f_1(x_1) + c_2 f_2(x_1) - y_1) f_1(x_1) + 2(c_1 f_1(x_2) + c_2 f_2(x_2) - y_2) f_1(x_2) + 2(c_1 f_1(x_3) + c_2 f_2(x_3) - y_1) f_1(x_3)$

Setting this to zero we get

$c_1(f_1(x_1)f_1(x_1) + f_1(x_2)f_1(x_2) + f_1(x_3)f_1(x_3)) + c_2(f_2(x_1)f_1(x_1) + f_2(x_2)f_1(x_2) + f_2(x_3)f_1(x_3)) = y_1 f_1(x_1) + y_2 f_1(x_2) + y_3 f_1(x_3)$

A similar equation results for $dE/dc_2 = 0$

Minimum squares: example

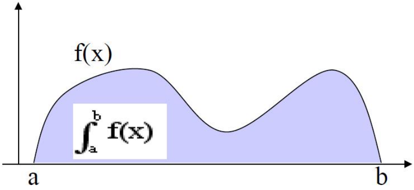
- Problem is that the eq. above is complicated
- Can be simplified by using vector notation and scalar product: let $\underline{x} = (x_1, x_2, x_3)$, and $\underline{y} = (y_1, y_2, y_3)$, then $\underline{x}.\underline{y} = x_1 y_1 + x_2 y_2 + x_3 y_3$
- Let $\underline{f}_1 = (f_1(x_1), f_1(x_2), f_1(x_3))$ and $\underline{f}_2 = (f_2(x_1), f_2(x_2), f_2(x_3))$
- The eq. become $c_1 \underline{f}_1.\underline{f}_1 + c_2 \underline{f}_1.\underline{f}_2 = y.\underline{f}_1$
  $c_1 \underline{f}_2.\underline{f}_1 + c_2 \underline{f}_2.\underline{f}_2 = y.\underline{f}_2$
- These equations can be solved with gaussian elimination
- Of course, this method can be extended to the general case

Minimum squares: general case

- Let us now examine the general case
  Problem: find the coefficients of
  $f(x) = c_1 f_1(x) + \ldots + c_M f_M(x)$
  at the obervation points $\underline{x} = (x_1, x_2, \ldots, x_N)$ $\underline{y} = (y_1, y_2, \ldots, y_N)$
- Let $\underline{f}_i = (f_i(x_1), \ldots, f_i(x_N))$ $(i = 1, \ldots, M)$
  and $\underline{f}_2 = (f_2(x_1), f_2(x_2), f_2(x_3))$
- Then one obtains a linear system of dimension M X M
  $A\underline{c} = \underline{b}$
- where $a_{ij} = \underline{f}_i.\underline{f}_j$ and $b_j = \underline{f}_y.\underline{y}$
- The solution of this eq. system delivers the coefficients seeked

Integration

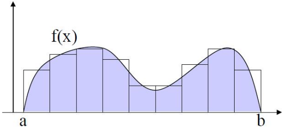
- Given a function, compute the integral of the function in a given interval



- Simplest way: compute symbolically the integral
- One gets the exact result
- Problem is: it cannot be always be done
  - Easy for polynomials, but not for complicated functions
- Symbolic manipulation packages, such as Maple, Mathematica or Maxima can do this

Simple quadrature methods

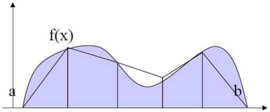
- The idea here is: evaluate function at regularly spaced points, then put rectangles around those points, and finally compute area of squares and add up



- Error (w interv. length, e depends on third deriv.):
$$\int_a^b f(x)d_x = r + w^3 e_3 + w^5 e_5 + \ldots$$
- So the resulting integral will be
$$r = \sum_{1 \leq i \leq N} (x_{i+1} - x_i) f(\frac{x_i + x_{i+1}}{2})$$
- Method not so bad at all: the integral if 1/x in [1,2] 10 0.6928353604100 100 0,6931440556283 1000 0,6931471493100 which makes it 7 decimals right for N=1000

Simple quadrature methods

- The idea here is: evaluate function at regularly spaced points, then put rectangles around those points, and finally compute area of trapezoids and add up



- Error: $\int_a^b f(x) d_x = t - 2w^3 e_3 + 4w^5 e_5 + \ldots$
- So the resulting integral will be
  $$r = \sum_{1 \leq i \leq N} (x_{i+1} - x_i) \frac{f(x_i) + f(x_{i+1})}{2}$$
- Method not so bad at all: the integral if 1/x in [1,2] 10 0.6937714031754 100 0,6931534304818 1000 0,6931472430599 which makes it 7 decimals right for N=1000

Simpson rule

- Combine both methods so as to eliminate first term in error.
- This is done by multiplying formula of rectangles by two, and adding the formula of trapeziods and then dividing results by 6
- Error: $\int_a^b f(x)d_x = 1/3(2r + t - 2w^5 e_5 + ...)$
- And the resulting formula is:
$$s = \sum_{1 \leq i \leq N} \frac{1}{6}(x_{i+1} - x_i)(f(x_i) + 4f(\frac{x_i + x_{i+1}}{2}) + f(x_{i+1}))$$
- Method much better: the integral if 1/x in [1,2] 10 0.6931473746652 100 0,6931471805795 1000 0,6931471805599