Collected Papers 2017

Mikko I. Malinen

ISBN 978-952-94-0038-6 (PDF)

Collected Papers 2017

Mikko I. Malinen

Publisher: Mikko I. Malinen

Joensuu, Finland

2018

If you cite this publication, you may do it by the following way:

Mikko I. Malinen, "Name of the article goes here", Collected Papers 2017, Mikko I. Malinen, Joensuu, Finland, ISBN 978-952-94-0038-6, 2018

Preface

This publication contains my articles from the year 2017. I thank Dr. Radu Mariescu-Istodor and Dr. Marcin Pilipczuk for reviewing the material. The line has been that almost all articles have had at least one reviewer.

Joensuu, Finland, 15th January, 2018

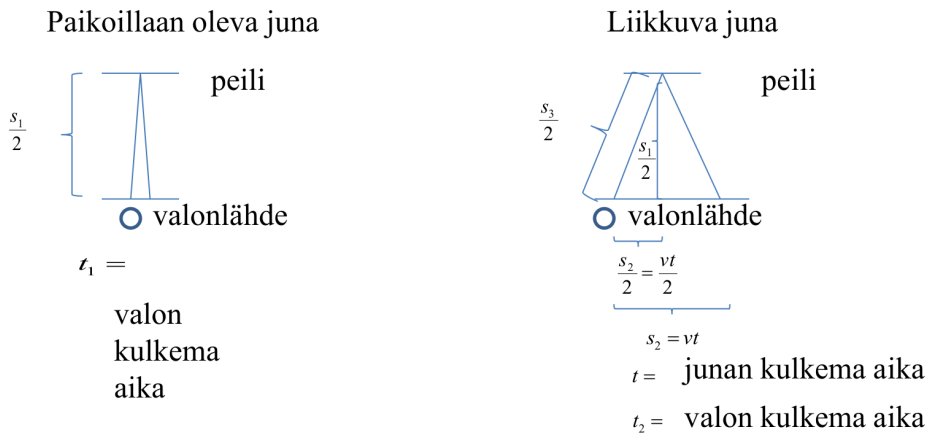
Dr. Mikko I. Malinen

Table of Contents

Suhteellisuusteorian aikadilataation johto

Mikko I. Malinen

13. tammikuuta 2018



Kuva 1. Valon kulku valonlähteestä havaitsijaan.

Johdamme seuraavassa suhteellisuusteorian aikadilataation lausekkeen

$$\frac{1}{\sqrt{1-(\frac{v}{c})^2}}.$$

Olettakaamme, että junan lattialle on asetettu valonlähde, jonka valo kulkee katossa olevan peilin kautta lattialla olevalle havaitsijalle (kuva 1). Kun juna on liikkeessä, joutuu valo kulkemaan pidemmän matkan (kuva 1 oik.). Laskemme kuinka paljon valon kulkema matka muuttuu suhteessa eli kuvan 1 merkinnöin suhteen s_3/s_1. Kuvan 1 asetelmista ja Pythagoraan lauseesta saadaan

$$(\frac{s_3}{2})^2 = (\frac{s_1}{2})^2 + (\frac{vt}{2})^2 = (\frac{s_1}{2})^2 + (\frac{vs_3}{2c})^2,$$

sillä $t_1 = \frac{s_1}{c}$ ja $t_2 = \frac{s_3}{c}$ ja $t_1 \neq t_2 = t$.

$$(\frac{s_3}{2})^2 = (\frac{s_1}{2})^2 + (\frac{vs_3}{2c})^2$$

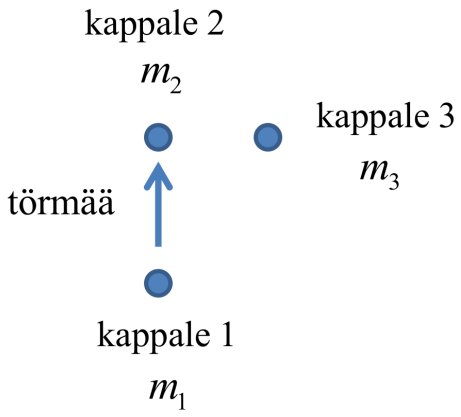
$$\frac{s_3^2}{4} - \frac{v^2 s_3^2}{4c^2} = (\frac{s_1}{2})^2$$
$$\frac{c^2 s_3^2}{4c^2} - \frac{v^2 s_3^2}{4c^2} = (\frac{s_1}{2})^2$$
$$\frac{s_3^2(c^2 - v^2)}{4c^2} = \frac{s_1^2}{4}$$
$$(\frac{s_3}{s_1})^2 = \frac{4c^2}{4(c^2 - v^2)}$$
$$(\frac{s_3}{s_1})^2 = \frac{c^2}{c^2 - v^2}$$
$$(\frac{s_1}{s_3})^2 = \frac{c^2 - v^2}{c^2}$$
$$= 1 - (\frac{v}{c})^2$$
$$\Rightarrow \frac{s_3}{s_1} = \frac{1}{\sqrt{1 - (\frac{v}{c})^2}},$$

joka on aikadilataatio.

Miksi Cernin bosonikoe ei aiheuttanut maapallon joutumista mustaan aukkoon?

Mikko Malinen

22. helmikuuta, 2017



Kuva 1. Kappaleiden 1 ja 2 törmäys.

Ennen Cernin bosonikoetta esitettiin arvelu, että koe muodostaisi mustan aukon, johon maapallo joutuisi. Osoitamme, että pienimassaisten kappaleiden törmäys ei muodosta mustaa aukkoa.

Kuvan 1 merkinnöin massan m_2 aiheuttama gravitaatiovoima kappaleeseen 3 on

$$F_{32_{alku}} = k \cdot \frac{m_2 \cdot m_3}{r^2}$$

Kun kappale 1 törmää kappaleeseen 2, niiden massat summautuvat:

$$F_{32_{loppu}} = k \cdot \frac{(m_2 + m_1) \cdot m_3}{r^2}.$$

Jos $m_1 = m_2$, niin F_{32} kasvaa kaksinkertaiseksi alku- ja lopputilanteen välillä. Tämä ei riitä synnyttämään mustaa aukkoa.

Finding the Closest Pair Between Two Linearly Separable Sets of Points

Mikko I. Malinen

24th February, 2017

Abstract

We show that finding the closest pair between two linearly separable set of points of sizes n and m in 2- or 3- dimensional Euclidean space takes $O(n + m)$ time complexity. This is an improvement to the brute force $O(nm)$ time complexity.

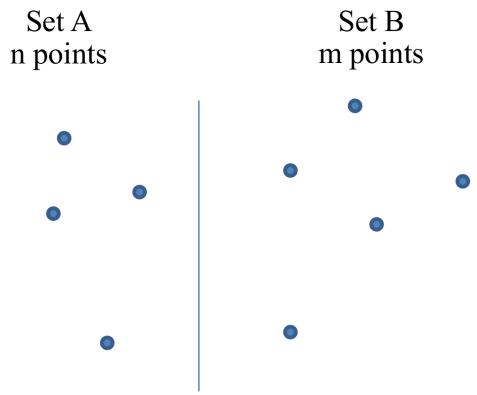


Figure 1. Two sets of points.

An example of two linearly separable sets of points is shown in Figure 1. To find the closest pair of points between the sets takes $O(nm)$ time by brute force calculation, where the distance of every pair is examined. The calculation has an application in an approximate minimum spanning tree (MST) construction. A divide-and-conquer technique divides a large dataset of points into smaller subsets, calculates exact MSTs within these subsets and combines the subsets by adding edges between them to form an approximate MST of the whole dataset [1]. See the Figure 2.

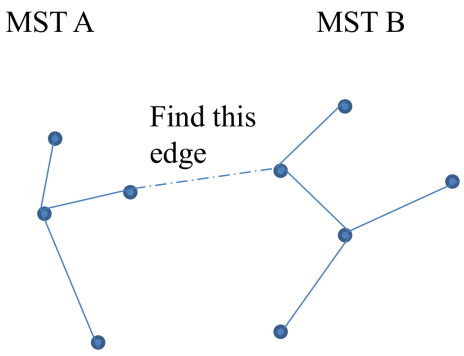


Figure 2. Two exact MSTs are combined into a bigger approximate MST.

Finding the closest pair may help also in constructing a support vector machine between sets of points. We use electrostatics to obtain an $O(n+m)$ time algorithm to find the closest pair. We connect the sets to a high-voltage source, so that there will be a voltage between the sets, see Figure 3.

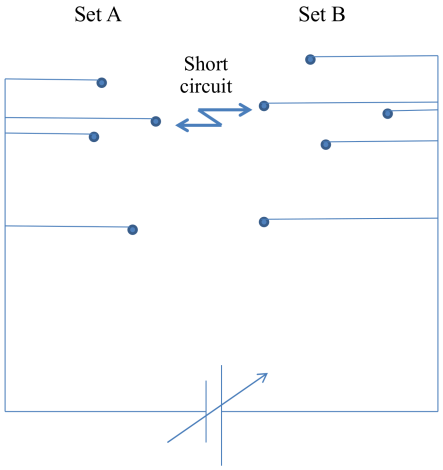


Figure 3. Electronic circuit to find the closest pair.

We increase the voltage, until a short circuit uccurs between the closest pair of points. We are interested in finding the pair, but the short circuit voltage is not important.

References

[1] Caiming Zhong, Mikko Malinen, Duoqian Miao and Pasi Fränti, "A Fast Minimum Spanning Tree Algorithm Based on k-Means", Information Sciences 295, February 2015, Pages 1-17

Approximate Sorting In Linear Time

Mikko I. Malinen

1st March, 2017

Abstract

We present an approximate sorting algorithm which works in linear $O(n)$ time. Some hardware is needed for the algorithm to work.

1 Literature

Paper [1] presents a different way to do sorting than with the conventional computer. Our approach is a computer algorithm and some hardware. In our approach, building the hardware takes linear time that is the bottleneck. All other phases take at most linear time.

2 Sorting equipment

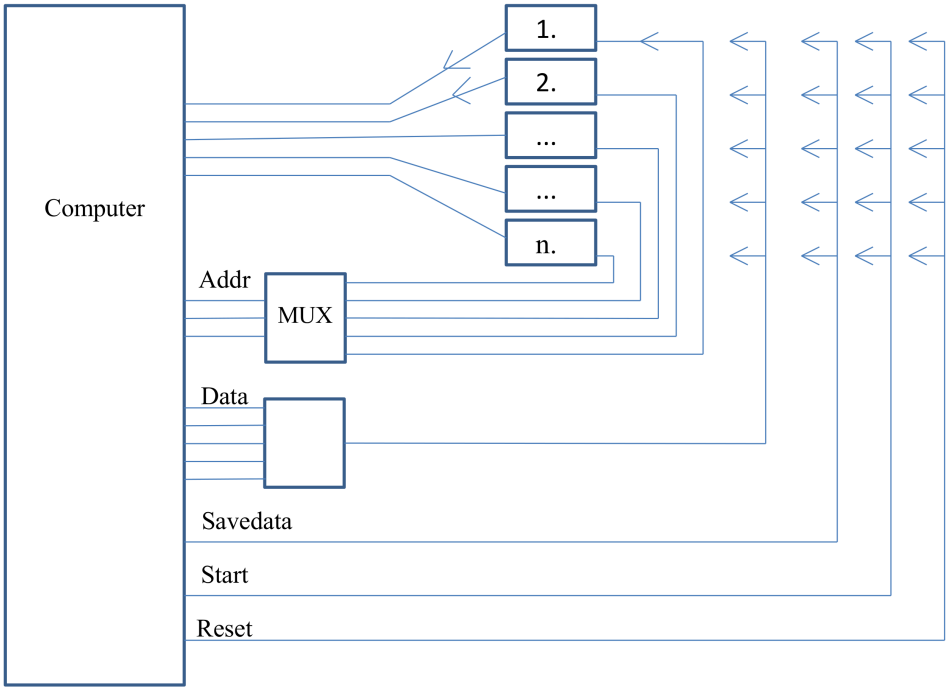


Figure 1. Sorting equipment.

The sorting equipment is shown in Figure 1. The timing information, "Data", is sent to each sending block 1.-n. on the right. The sending block number where the timing information is to be sent is determined by the address, "Addr" in the figure, which is a binary number and after the multiplexer, "MUX" in the Figure, it is converted to bit value one for the block corresponding to the address. "Savedata" bit triggers the save. When the timing information is saved, the sending blocks 1.-n. on the right send "1", each at a time determined by the dataset item value 1.-n., correspondingly. The computer scans the inputs in a round robin manner, i.e. circulating, reading one input at a time. If the computer reads "1", it saves the corresponding data item to its sorted place in an array, and resets this input.

3 Algorithm

Phases and time complexities of them in sorting are:

Send timing information to n blocks $O(n)$.

Trigger start
REPEAT Scan outputs from blocks by round robin manner $O(n)$
-if state "1" found
-store corresponding value.
-set input to "0".
UNTIL maximum time reached.

4 Accuracy Analysis

Let the minimum and maximum values in the data be x_{min} and x_{max}. Let the speed of the scanner be m values/s. Then one value is checked in time 1/m seconds. All are checked (round done) in time $t_{allcheck} = n \cdot 1/m$ seconds. All are sent in time $t_{allsent}$, which is a constant. Amount of range sent in one second is $r_s = (x_{max} - x_{min})/t_{allsent}$. Accuracy of sort is then

$$A = r_s \cdot t_{allcheck} = \frac{x_{max} - x_{min}}{t_{allsent}} \cdot \frac{n}{m} \cdot 1s.$$

The value of A is smaller when the sort is more accurate. A tells how much one single sorted element may differ from value of an element in the same position in exact sort.

References

[1] A. K. Dewdney, "On the spaghetti computer and other analog gadgets for problem solving", Scientific American, 250 (6), pp. 1926, June 1984

The SAT Problem is not harder than A Zero of a Multivariate Expression

Mikko Malinen

13th May, 2017

Abstract

We show that a Satisfiability (SAT) problem instance can be represented as a multivariate expression so that solving a zero of that expression gives also a solution to the SAT problem instance.

1 Introduction

A SAT problem instance is a conjunction ($\wedge$) of clauses which are disjunction ($\vee$) of propositions or propositions' negations ($\neg$). A solution of a SAT instance is a truth assignment which makes the instance true.

2 Representing a SAT Instance as a Multivariate Expression

We make correspondences as in Table 1. For example, the SAT instance

Table 1: Correspondences between SAT instance and multivariate expression

SAT	Expression	Becomes zero when
$A \wedge B$	$A + B$	A and B are zero
$A \vee B$	$(1 - A) \cdot (1 - B)$	A or B is 1
$A \vee \neg B$	$(1 - A) \cdot B$	A is 1 or B is zero
$\neg A \vee \neg B$	$A \cdot B$	A is zero or B is zero
$\neg A$	A	A is zero

$$(A \vee \neg B) \wedge (\neg A \vee \neg B)$$

would become a multivariate expession

$$(1 - A) \cdot B + A \cdot B$$

We need to solve a zero of the expression in range $A \in [0,1], B \in [0,1]$ to get a solution of the SAT instance. Variable values 0 and 1 are the truth assignment (corresponding FALSE and TRUE, and variable values which are between 0 and 1 we don't care (those have no effect to the truth value of the SAT instance). We conclude that solving a SAT instance is not harder than solving a zero of a multivariate expression.

Mean Squared Error Equals Generalized Variance

Mikko Malinen

25th May, 2017

Abstract

Variance is defined for one-dimensional data. We generalize variance to multidimensional data by taking the sum of variances along different dimensions. We show that it equals mean squared error (MSE). We use this result to prove that scaling a dataset b factor a, MSE changes by factor a^2. We show that in one-dimensional data, data items can be replaced by moments and still the data can be recovered.

1 Generalized Variance

Variance is defined for one-dimensional data:

$$X = \{x_1, x_2, ..., x_n\}$$

$$Var = \mathrm{E}[X^2] - (\mathrm{E}[X])^2.$$

We ask, can variance be generalized to multidimensional data? And can MSE of a dataset be equal to this generalized variance measure? Consider a several-dimensional dataset as in Figure 1, where two dimensions are shown. Variances along different dimensions are

$$\begin{aligned} Var_1 &= \mathrm{E}[x_{i1}|1 \leq i \leq n] - (\mathrm{E}[x_{i1}|1 \leq i \leq n])^2 \\ &= \frac{x_{11} + x_{21} + ... + x_{n1}}{n} - m_1^2 \\ Var_2 &= \mathrm{E}[x_{i2}|1 \leq i \leq n] - (\mathrm{E}[x_{i2}|1 \leq i \leq n])^2 \\ &= \frac{x_{12} + x_{22} + ... + x_{n2}}{n} - m_2^2 \end{aligned}$$

...

$$Var_d = ...$$

Let's move the centroid to origo and points an equal amount towards origo. That does not affect variances.

$$Var_1 = \frac{(x_{11} - m_1)^2 + (x_{21} - m_1)^2 + ... + (x_{n1} - m_1)^2}{n}$$

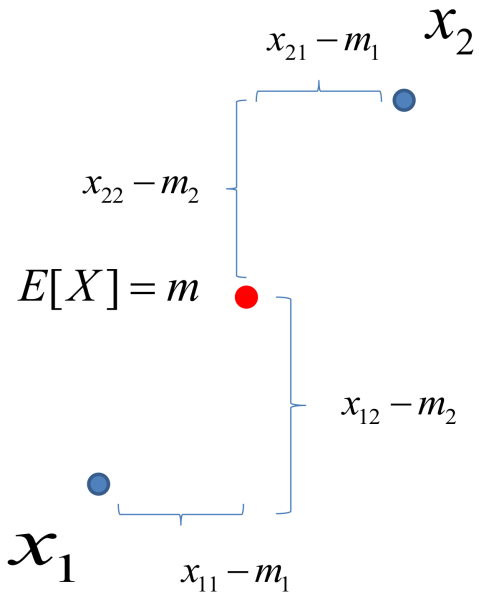


Figure 1: A dataset of several dimensions.

$$Var_2 = \frac{(x_{12} - m_2)^2 + (x_{22} - m_2)^2 + ... + (x_{n2} - m_2)^2}{n}$$

...

$$Var_d = ...$$

Now let's define *generalized variance* to be the sum of individual variances.

$$GenVar = Var_1 + Var_2 + ... + Var_d$$

$$= \frac{(x_{11} - m_1)^2 + (x_{21} - m_1)^2 + ... + (x_{n1} - m_1)^2}{} ...$$

$$\frac{+(x_{12} - m_2)^2 + (x_{22} - m_2)^2 + ... + (x_{n2} - m_2)^2}{} ...$$

...

$$\frac{+(x_{1d} - m_d)^2 + (x_{2d} - m_d)^2 + ... + (x_{nd} - m_d)^2}{n}$$

MSE equals

$$MSE = \sum_i \sum_s (x_{is} - m_s)^2/n, \quad 1 \leq i \leq n \quad 1 \leq s \leq d,$$

and we denote

$$MSE_s = \sum_i (x_{is} - m_s)^2/n, \quad 1 \leq i \leq n \quad 1 \leq s \leq d.$$

This implies that

$$GenVar = MSE_1 + MSE_2 + ... + MSE_d$$

$$= \frac{||x_1 - m||^2 + ||x_2 - m||^2 + ... + ||x_n - m||^2}{n}$$

$$= MSE.$$

Since

$$GenVar = Var_1 + Var_2 + ... + Var_d = MSE = \text{constant}$$

independent of the orientation of the orthogonal axes,

$$Var_1 + Var_2 + ... + Var_d = \text{constant}$$

independent of the orientation of the orthogonal axes.

1.1 Application in Principal Component Analysis

When we have the principal component analysis result (variances) available (see Figure 2, to calculate MSE of the desired dimensions, it is adequate to sum the variances of those dimensions.

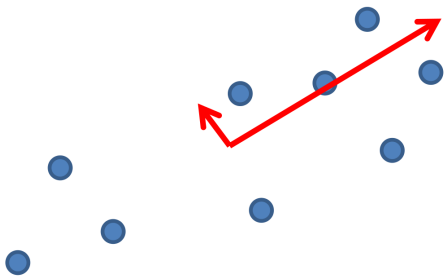


Figure 2: In PCA directions of largest variances are found.

2 Scaling a Dataset

Formula for variance is

$$Var = \mathrm{E}[X^2] - (\mathrm{E}[X])^2.$$

When scaling data by factor a, the variance changes by factor a^2:

$$Var_a = \mathrm{E}[(aX)^2] - (\mathrm{E}[aX])^2$$

$$= a^2\mathrm{E}[X^2] - a^2(\mathrm{E}[aX])^2 = a^2 \cdot Var$$

That is MSE changes by factor a^2. This is true also for multidimensional data, since the individual variances along different dimensions sum up to MSE.

3 Replacing Data by Moments

Consider a one-dimensional dataset, see Figure 3. 1st moment is the mean

$$\mathrm{E}[X] = \frac{x_1 + x_2}{2} = b$$

By knowing the 1st moment's value one doesn't need to know the value of one point and still reconstruct the dataset. 3rd moment is

$$\mathrm{E}[X^3] = \frac{x_1^3 + x_2^3}{2} = c.$$

By knowing the 1st and 3rd moments' values one doesn't need to know the value of two points and still reconstruct the dataset (two unknown variables and two equations. Let's write the equations into simpler form:

$$x_1 + x_2 = d$$

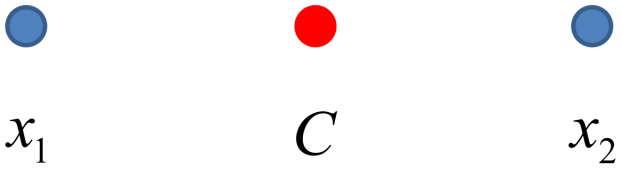


Figure 3: A one-dimensional dataset.

$$x_1^3 + x_2^3 = f$$

By increasing the number of moments, less point information is needed. In fact, we need to know n of the following:

$$\text{Data } x_i, \quad i = [1..n] \tag{1}$$
$$\text{Moments } \mathrm{E}[X^s], s = 1, 3, 5, ... \tag{2}$$

to reconstruct the data. If we know n of the odd moments, no data is needed. By using odd moments, we get the unique solution. If we used even moments, there would be a danger to get non-unique solutions, since a value and its negative could both be solutions.

If we assume the data to be nonnegative, we can use any moments to get the unique solution. We need to know n of the following:

$$\text{Data } x_i, \quad i = [1..n] \tag{3}$$
$$\text{Moments } \mathrm{E}[X^s], s = [1..n] \tag{4}$$

to reconstruct the data. If we know n of the moments, no data is needed.

This can be done separately for each dimension, but we must note that permutation of data features is not solved. For example, data could be after reconstruction
X features 1, 2
Y features 3, 4
This can be interpreted as dataset

$$\{(1,3),(2,4)\} \quad \text{or} \quad \{(1,4),(2,3)\}$$

and we have no way to know which one is correct.

4 Conclusions

We have defined a generalized variance, which equals MSE. We have used that to show how a scaling of dataset affects MSE. We have shown that moments can replace data in one-dimensional datasets.

In High Dimensions, Squared Distances Start to Obey Normal Distribution

Mikko Malinen

27th May, 2017

Abstract

We show that in distributions which are identical in each dimension the squared pairwise distances start to obey normal distribution when dimensionality approaches infinity.

1 Special Result

If we represent two points as vectors $x_i = [x_{i1}x_{i2}...x_{id}], x_j = [x_{j1}x_{j2}...x_{jd}]$, then the Euclidean distance between any pair of points is

$$dist_{ij} = \sqrt{|x_{i1} - x_{j1}|^2 + |x_{i2} - x_{j2}|^2 + ... + |x_{id} - x_{jd}|^2}$$

$$= \sqrt{dfeat_1^2 + dfeat_2^2 + ... + dfeat_d^2}, \qquad dfeat_s \geq 0, \quad dfeat_s^2 \geq 0 \quad \forall s = 1..d.$$

We assume that the $dfeat_s$s are independent and identically distributed. This can be achieved, when the vectors are randomly distributed into a suitable restricted domain in space. This domain can be for example a hypercube. Then the $dfeat_s^2$s are also independent and identically distributed. By the central limit theorem [1] when the dimensionality is high, the distribution of the $dist_{ij}^2$s is approximately normal, and thus the distribution of the $dist_{ij}$s is approximately the square root of normal (skewed normal). To proceed further, we make the hypothesis that in high dimensions the clustering results (costs) both in cost function types $dist_{ij}^2$ and $dist_{ij}$ do not differ as much as in low dimensions when the centroid locations C change, i.e., there may be nearer results (cost) for centroid locations that are far away or partitions that are very different than in low dimensions.

2 General Result

In Theorem 1 we state a result, which is valid for more general distributions than in the preceding text.

Theorem 1. Let vectors be independent and distributed identically in every dimension. Then their endpoint's squared pairwise distances approach normal distribution when the dimensionality increases.

Proof. Because vectors are independent, their endpoints' pairwise distances per dimension ($dfeat_s$s) are also independent. The rest similarly using central limit theorem as in the preceding text.

3 Conclusions

We have proven that for independent vectors and their disributions which are identical in every dimension, the squared pairwise distances start to obey normal distribution when the dimensionality increases. However, it has to be noted, that in practice, data is often correlated and in different scales in different dimensions, that is, the data is not often independent nor identically distributed among features.

References

[1] J. Rice. *Mathematical statistics and data analysis (Second ed.)*. Duxbury Press, 1995.

Time Complexity T of a Program Cannot Be Decided in Time Complexity T

Mikko Malinen

28th May, 2017

Abstract

We prove that the time complexity T of a given program, which solves some problem with any input size cannot be decided in time complexity T. It is proved by a modification of a proof of unsolvability of the halting problem.

1 Introduction

Determining the time complexity is an important task in algorithm design and analysis. Polynomial or less time complexity algorithms are considered tractable and exponential or more time complexity algorithms are considered intractable. We deal with *programs*, which are implementations of algorithms and which have the same time complexity as the corresponding algorithms.

We are going to prove that time complexity T of a program cannot be proven in time complexity T. For this we are going to use a modification of the halting problem [1]. We review here a couple of problems, which are related to the halting problem.

The truth value of a "theorem" referring to itself [2]

Theorem 1. This theorem is false.

cannot be determined since if it is true, it is false as the theorem says and if it is false, it is true because it is not false. Another related theorem is the Gödel's incomleteness theorem [2], for which the proof is based on similar technique as the Theorem 1. [2]. From Gödel's incomleteness theorem follows that it is not possible to construct a complete theorem prover which would cover everything, because to be able to find all theorems, the theory would need to be extended infinitely.

The halting problem proof is constructed so that if the given program halts, it does not halt and if it does not halt it halts. This makes the problem undecidable.

2 Undecidability

It is a well-known fact that in general it is undesidable whether a given computer program halts or not. Here we consider only programs that will halt. We prove that in general it is not possible to prove that a program is of time complexity T in the same time complexity T, where $T = O$(some expression). We separate input of a program and a program. We consider time complexity with respect to size of input+program. For a large input size the input+program size is almost the same as the input size, for our purpose there is no effective difference. For example, for a traveling salesman problem (TSP) program the input could start

d(1,1)=0
d(1,2)=3
d(1,3)=23,

where the distances between the cities are given. We assume that we have a function is_T(program,input), which decides is the program of time complexity T. And we assume that this function halts in time T with respect to input+program size. We need also a function execute_longer_than_T(input), which does not output any useful information, but it executes longer than T time. We write also an other program, which utilizes the is_T() -function:

program P()
input Q
if is_T(Q,NULL)="yes" execute_longer_than_T(Q)
end

As in halting problem proof, we give this program itself as input to is_T -function. If is_T -function halts in time T, as was the assumption, and gives answer "yes", then the program will execute longer than time T. But if the is_T() -function gives answer "no", then the program P will halt in time T. Both of these cases are contradictory, so we can deduct, that is_T() can not generally be decided in time T.

3 Conclusion

We proved that time complexity T of a program cannot be decided in time complexity T, where $T = O$(some expression). This we did by a modification of the halting problem proof.

References

[1] J. G. Brookshear. *Computer Science An Overview, Eleventh Edition*. Pearson, 2012.

[2] J. Väänänen. *Matemaaattinen logiikka*. Gaudeamus, 1987.

On Gravitational Waves

Mikko I. Malinen

28th August, 2017

1 Introduction

Two stars that circle around each other cause gravitational waves, that is, the gravitational force they affect to an observer is not constant. The force is maximal when the observer and the stars are on the same line. We show that when the stars are of unequal mass, the maximum force happens when the smaller star is in its nearest position relative to the observer on that line.

2 Theory

Consider the case 1 where the smaller star is in its nearest position in Figure 1 and the case 2 where the bigger star is in the nearest position in Figure 2. We denote the mass of the bigger star by M and the mass of the smaller star by aM, $0 \leq a < 1$, and the mass of the observer by m. In Figures 1 and 2 distances are shown and calculated.

Gravitational force can be calculated by the formula

$$F = k \cdot \frac{m_1 \cdot m_2}{d^2}, \tag{1}$$

where m_1 and m_2 are the masses of the objects, d is the distance between the objects and k is the gravitational constant. From (1) and Figure 1 we get the formula for the force affecting the observer in case 1:

$$F_1 = k \cdot \frac{m \cdot aM}{(d_2 - \frac{M}{aM+M} \cdot d_1)^2} + k \cdot \frac{m \cdot M}{(d_2 + \frac{aM}{aM+M} \cdot d_1)^2}. \tag{2}$$

From (1) and Figure 2 we get the formula for the force affecting the observer in case 2:

$$F_2 = k \cdot \frac{m \cdot aM}{(d_2 + \frac{M}{aM+M} \cdot d_1)^2} + k \cdot \frac{m \cdot M}{(d_2 - \frac{aM}{aM+M} \cdot d_1)^2}. \tag{3}$$

Without loss of generality we can set $a = 0.5$ for comparing F_1 and F_2:

$$F_1 = k \cdot \frac{m \cdot 0.5M}{(d_2 - \frac{1}{1.5} \cdot d_1)^2} + k \cdot \frac{m \cdot M}{(d_2 + \frac{0.5}{1.5} \cdot d_1)^2} \tag{4}$$

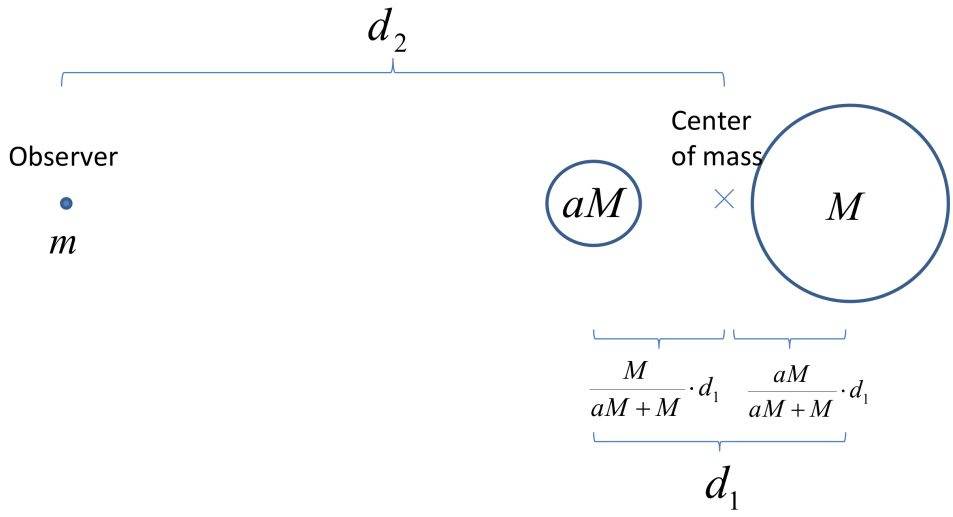


Figure 1: Setting where the smaller star is in its nearest position relative to the observer.

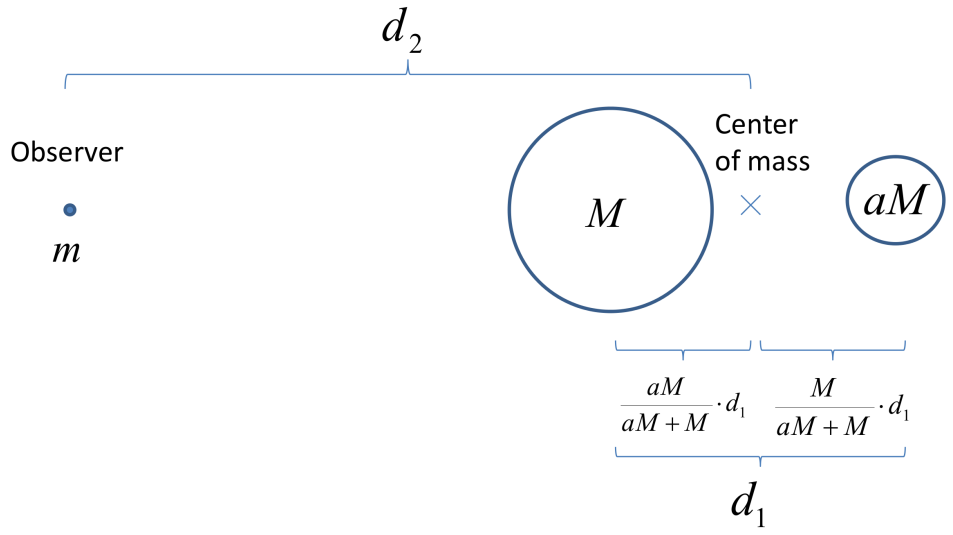


Figure 2: Setting where the bigger star is in its nearest position relative to the observer.

$$F_2 = k \cdot \frac{m \cdot 0.5M}{(d_2 + \frac{1}{1.5} \cdot d_1)^2} + k \cdot \frac{m \cdot M}{(d_2 - \frac{0.5}{1.5} \cdot d_1)^2} \tag{5}$$

Without loss of generality we can set $d_2 = 2 \cdot d_1$ and $m = M = 1$ for comparing F_1 and F_2:

$$F_1 = k \cdot \frac{0.5}{(0.5d_1)^2} + k \cdot \frac{1}{(d_1)^2} \tag{6}$$

$$F_2 = k \cdot \frac{0.5}{(\frac{2.5}{1.5}d_1)^2} + k \cdot \frac{1}{(\frac{1}{1.5}d_1)^2} \tag{7}$$

Without loss of generality we can set $d_1 = 1$:

$$F_1 = k \cdot 2 + k \cdot 1 = 3k \tag{8}$$

$$F_2 = k \cdot \frac{0.5}{\frac{25}{9}} + k \cdot \frac{1}{\frac{4}{9}} = k \cdot \frac{4.5}{25} + k \cdot \frac{9}{4} = k \cdot \frac{18 + 225}{100} = k \cdot \frac{243}{100} \tag{9}$$

so the force is bigger in case 1.

Number of Operations in Distance Calculation

Mikko I. Malinen

29th August, 2017

1 Introduction

Euclidean squared distance between two points $x_1 = \{x_{11}, x_{12}, ..., x_{1d}\}$ and $x_2 = \{x_{21}, x_{22}, ..., x_{2d}\}$ can be calculated by

$$(x_{11} - x_{21})^2 + (x_{12} - x_{22})^2 + ... + (x_{1d} - x_{2d})^2 \tag{1}$$

We are interested in is this the way to calculate it with the least number of operations. We know that it has $2d$ variables, so it will need at least $2d - 1$ operations (one variable needs at least one operation except the first variable).

2 Theory

Let's focus on the two-dimensional case without loss of generality. The distance is

$$(x_{11} - x_{21})^2 + (x_{12} - x_{22})^2. \tag{2}$$

All $x_{ij} : s$ are independent so we can write it as

$$a^2 + b^2. \tag{3}$$

This requires three operations (two squares and one addition). Two basic operations would lead to expressions like $a^2 + b$, $ab - b$ etc. We see that the distance cannot be calculated with two operations and so (1) is the expression with least number of operations.

Possible Parallel Universes

Mikko I. Malinen

2nd September, 2017

Abstract

We show one possibility for parallel universes. In this example there exists four universes: one by the three dimensions that are visible to us and one parallel universe for each of these three dimensions. We may exist simultaneously in all of these universes.

1 Result on Euclidean Distance

We begin by showing that Euclidean two-dimensional distance $\sqrt{a^2+b^2}$, where a and b are distances along different coordinates, cannot be presented as $f(a)+g(b)$, where f and g are functions depending only on one of these coordinates. This is due to the fact that contribution of b depends on a and contribution of a depends on b. This result generalizes to higher dimensions also.

2 Parallel Universes

Euclidean distance in our universe that we see can be calculated as $\sqrt{a^2+b^2+c^2}$. It is possible, that each of these dimensions have coresponding parallel multi-dimansional universe, for which the distance moved is the same as the distance moved along the corresponding coordinate a, b or c, but the distance is moved in higher dimensional space for which the move along coordinate i are presented by a_i, b_i and c_i in the following way:

$$a^2 = a_1^2 + a_2^2 + ... + a_{max}^2 \tag{1}$$

$$b^2 = b_1^2 + b_2^2 + ... + b_{max}^2 \tag{2}$$

$$c^2 = c_1^2 + c_2^2 + ... + c_{max}^2. \tag{3}$$

We see that each of these equations are equations of a hyperball, where the radii are a, b and c. So if we move for example distance a along the first dimension, in the first parallel universe we may end up to some point of a hyperball, which radius is a. There is also freedom to choose the orientation of the axes in the ordinary three-dimensional universe, so moving distance s may result in movement of $0 \leq x \leq s$ along certain dimension, so in the corresponding parallel universe we may end up also to inside the hyperball.