NATIONAL AND KAPODISTRIAN UNIVERSITY OF ATHENS
TRAINING AND LIFELONG LEARNING CENTER

Problem Solving with Artificial Intelligence and Advanced Estimation Algorithms

Exercises

Educational Module 1: Uncertainty, Search, Knowledge, Learning

ASPROUDIS IASONAS-CHRISTOPHOROS

Athens December 2022

Table of Contents

1. SEARCH FOR APPS BASED ON THE SAME PRINCIPLES

1A) Search the Internet: An improved version of the (least squares) method for non-linear models (see Marquardt) and a method for stochastic models.

Improved version for non-linear models

i. **Non-linear least squares** is the form of least squares analysis used to fit a set of m observations with a model that is non-linear in n unknown parameters ($m \geq n$). It is used in some forms of nonlinear regression. The basis of the method is to approximate the model by a linear one and to refine the parameters by successive iterations. There are many similarities to linear least squares, but also some significant differences. In economic theory, the non-linear least squares method is applied in (i) the probit regression, (ii) threshold regression, (iii) smooth regression, (iv) logistic link regression, (v) Box-Cox transformed regressors.

   Some nonlinear regression problems can be moved to a linear domain by a suitable transformation of the model formulation.

   However, use of a nonlinear transformation requires caution. The influences of the data values will change, as will the error structure of the model and the interpretation of any inferential results. These may not be desired effects. On the other hand, depending on what the largest source of error is, a nonlinear transformation may distribute the errors in a Gaussian fashion, so the choice to perform a nonlinear transformation must be informed by modeling considerations.

ii. **The Gauss–Newton algorithm** is used to solve non-linear least squares problems, which is equivalent to minimizing a sum of squared function values. It is an extension of Newton's method for finding a minimum of a non-linear function. Since a sum of squares must be nonnegative, the algorithm can be viewed as using Newton's method to iteratively approximate zeroes of the sum, and thus minimizing the sum. It has the advantage that second derivatives, which can be challenging to compute, are not required.[1]
   Non-linear least squares problems arise, for instance, in non-linear regression, where parameters in a model are sought such that the model is in good agreement with available observations.
   The method is named after the mathematicians Carl Friedrich Gauss and Isaac Newton, and first appeared in Gauss' 1809 work Theoria motus corporum coelestium in sectionibus conicis solem ambientum.

iii. In mathematics and computing, **the Levenberg–Marquardt algorithm** (LMA or just LM), also known as the damped least-squares (DLS) method, is used to solve non-linear least squares problems. These minimization problems arise especially in least squares curve fitting. The LMA interpolates between the Gauss–Newton algorithm (GNA) and the method of gradient descent. The LMA is more robust than the GNA, which means that in many cases it finds a solution even if it starts very far off the final minimum. For well-behaved functions and reasonable starting parameters, the LMA tends to be slower than the GNA. LMA can also be viewed as Gauss–Newton using a trust region approach.

The algorithm was first published in 1944 by Kenneth Levenberg,[2] while working at the Frankford Army Arsenal. It was rediscovered in 1963 by Donald Marquardt,[3] who worked as a statistician at DuPont, and independently by Girard,[4] Wynne[5] and Morrison.[6]
The LMA is used in many software applications for solving generic curve-fitting problems. By using the Gauss–Newton algorithm it often converges faster than first-order methods.[7] However, like other iterative optimization algorithms, the LMA finds only a local minimum, which is not necessarily the global minimum.

Improved version for stochastic models

iv. In estimation theory, the **extended Kalman filter (EKF)** is the nonlinear version of the Kalman filter which linearizes about an estimate of the current mean and covariance. In the case of well defined transition models, the EKF has been considered[8] the de facto standard in the theory of nonlinear state estimation, navigation systems and GPS.[9]

The Kalman filter, under Gaussian assumption, is the optimal state estimator for linear dynamic systems. Although originally devised for linear systems, nonlinear systems can also be addressed by the Kalman filter through some modifications to it as approximations to the optimal state estimator. The extended Kalman filter (EKF) is one of the most popular estimation techniques that has been largely investigated for state estimation of nonlinear systems. The EKF uses the standard Kalman filter equations to the first-order approximation of the nonlinear model about the last estimate. It is very sensitive to initialization, and filter divergence is inevitable if the arbitrary noise matrices have not been chosen appropriately.

Recently there has been an increased research attention for Networked Control Systems. A common feature of these systems is the presence of significant communication delays and data loss across the network. Therefore, it has become necessary to jointly address the issues of control and communication in these systems. In contrast to traditional filtering problems, an important feature in networked systems is that the delivery of measurements to the estimator is not always reliable and losses of data may occur. This leads to estimation schemes which are required to handle missing data. For example, Figure 1 shows a structure where the arrival of an observation is modelled by a binary stochastic variable γ_t. If a measurement arrives after the t^{th} step, γ_t is set to 1, if no measurement arrives after the t^{th} step, γ_t is set to 0. The stability and convergence properties of the estimation process have been studied in the case of linear Kalman filtering with intermittent observations, where it is shown that there exists a certain threshold of the packet loss rate above which the state estimation error diverges in the expected sense, i.e., the expected value of the error covariance matrix becomes unbounded as time goes to infinity. The lower and upper bounds of the threshold value is also provided.

***1B) Search the Internet**: What is ACAS and how does it work for aircraft or autonomous vehicles. Also look for a recent variant of it that applies AI to protect passing birds from wind turbines (Greek company/patent)?*

Traffic Alert and Collision Avoidance System (TCAS) is an implementation of the ICAO Airborne Collision Avoidance System (ACAS) standard. In fact, it's currently the only implementation of the ACAS so the two terms, TCAS and ACAS, are often used interchangeably

ACAS/TCAS can issue two types of alerts: Traffic advisories (TAs), which aim to help the pilots in the visual acquisition of the intruder aircraft, and to alert them to be ready for a potential resolution advisory. Resolution advisories (RAs), which are avoidance manoeuvres recommended to the pilot.

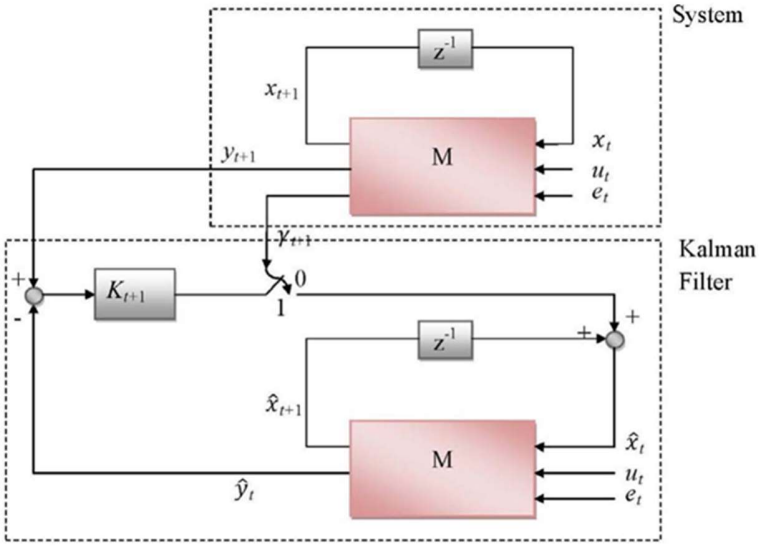


Figure 1. An estimation scheme for Kalman filtering with measurement loss

"Different colours, patterns, or less-conspicuous approaches like using UV-reflective paint could also be promising ways to reduce bird mortality. Above all, **siting turbines away from high bird-use areas** remains the best way to reduce bird mortality."

MUSE AI uses artificial intelligence (AI) and video analysis and enables the operation of wind turbines to be adjusted when specific bird species area present in order to protect the birds.

MUSE is already in use, integrated offshore windfarms' control systems, but has been enhanced with the addition of AI-based species recognition, automating the process and enabling speed reductions on wind turbines when specific bird species fly through a windfarm.

DHI says AI-based species recognition enables the shutdown of individual wind turbines. The AI functionality enables highly accurate species recognition and algorithms calculate birds' route through a windfarm. As a result, automatic shutdown can therefore be limited to a single turbine, ensuring more efficient energy production.

DHI executive vice president for energy and ports Mikael Kamp Sørensen said, “MUSE AI is an example of how we can apply innovation and 30 years of experience with offshore wind to ensure more renewable energy and protection of biodiversity. Sustainable development is at the core of our business in DHI, and we are proud to be able to bring MUSE AI to the market.”

MUSE AI has been intensively tested at DHI’s test facilities in Denmark, which will also act as a demonstration centre for the technology.

Within the context of its participation (Hall 3, Stand D30) in the leading Annual European Wind Energy Exhibition, organised by Wind Europe (Pan-European Wind Energy Association) between 5-7 April in Bilbao, Spain, Nvisionist will be joining the upcoming session “Technology for mitigating and ensuring positive biodiversity impacts” organized by Wind Europe which will be held on Wednesday 6th April 2022 at 16:15-17:15, WindTalks Stage for Innovation (Exhibition hall 1).

Alexander Vandenberghe, Sustainability Manager at WindEurope, will be moderating the session while highly esteemed experts on wind energy will also participate: Hywel Roberts, Senior Lead Strategic Specialist, Ørsted, Ibon Galparsoro, Prinicpal Researcher, AZTI, Tassos Alefantos, CEO, Nvisionist and Cristina Simioli, Senior Manager - Energy and Policy Systems, Renewables Grid Initiative. Moreover, Mr Tassos Alefantos, CEO of Nvisionist, will be presenting for the first time nvbird® - offshore, Nvisionist’s new product. Nvbird® - offshore is a unique & innovative product designed especially for offshore Wind Turbine Generators, adding value to the wind energy by almost eliminating the erroneous shutdown of offshore wind turbines, maximizing their productivity, while safeguarding protected birds from colliding with wind turbines’ blades.

nvbird® is a pioneering, integrated solution with application in Wind Energy and with significant benefits for the environment. It is at the forefront of global technology by using state-of-the-art artificial intelligence and machine learning technologies while at the same time incorporating the latest hardware, software as well as business intelligence platforms.

It is based on a unique algorithm allowing the system to detect and recognize with unprecedented accuracy birds that fly dangerously near the wind turbines, analyzes their flight path, activates speakers with sounds to deter them and in case they do not fly away it stops the wind turbine until the birds are safe.

Nvisionist is an innovative high-tech start-up that specializes in applied digital technology solutions, based on Artificial Intelligence (AI) and Machine Learning. Nvisionist designs, creates and offers innovative solutions and services for the renewable energy sector. Nvisionist design implementation, offers benefits to organizations, communities, protects the environment, and contributes both to the quality of life and conservation of resources

1C) Search the Internet for*: Advanced Ai applications relevant to your specialty and provide a brief description of them.*

Recommendation System

Various platforms that we use in our daily lives like e-commerce, entertainment websites, social media, video sharing platforms, like youtube, etc., all use the recommendation system to get user data and provide customized recommendations to users to increase engagement. This is a very widely used Artificial Intelligence application in almost all industries.

Applications of Artificial Intelligence in Social Media

Instagram

On Instagram, AI considers your likes and the accounts you follow to determine what posts you are shown on your explore tab.

Facebook

Artificial Intelligence is also used along with a tool called DeepText. With this tool, Facebook can understand conversations better. It can be used to translate posts from different languages automatically.

Twitter

AI is used by Twitter for fraud detection, removing propaganda, and hateful content. Twitter also uses AI to recommend tweets that users might enjoy, based on what type of tweets they engage with.

Applications of Artificial Intelligence in Marketing

Artificial intelligence (AI) applications are popular in the marketing domain as well.

- Using AI, marketers can deliver highly targeted and personalized ads with the help of behavioral analysis, and pattern recognition in ML, etc. It also helps with retargeting audiences at the right time to ensure better results and reduced feelings of distrust and annoyance.
- AI can help with content marketing in a way that matches the brand's style and voice. It can be used to handle routine tasks like performance, campaign reports, and much more.
- Chatbots powered by AI, Natural Language Processing, Natural Language Generation, and Natural Language Understanding can analyze the user's language and respond in the ways humans do.
- AI can provide users with real-time personalizations based on their behavior and can be used to edit and optimize marketing campaigns to fit a local market's needs.

https://www.simplilearn.com/tutorials/artificial-intelligence-tutorial/artificial-intelligence-applications

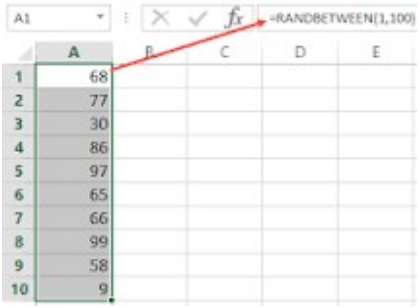
2. UNCERTAINTY PROBABILITY DISTRIBUTIONS & RANDOM NUMBERS

***2A) Search the Internet**: Random Number Tables. Check if they are also in .csv format to be read by PC programs. List the sources/links.*

How do I generate a random number in a CSV file?
I suggest:

1. open CSV file using Excel Command.
2. Generate a random number between 1 and total number of rows and save it in $vRandomNumber$
3. Use Get Cell Command > Specific Cell > AvRandomNumber$

How do I generate 10000 random numbers in Excel?



Here are the steps to generate random numbers in Excel without repetition:

1. Select the cells in which you want to get the random numbers.
2. In the active cell, enter =RAND()
3. Hold the Control key and Press Enter.
4. Select all the cell (where you have the result of the RAND function) and convert it to values.

https://trumpexcel.com/generate-random-numbers-excel/

***B) Search the Internet**: Random Number Generators of various formats and for different distributions. List the apps and what kind they are (online, download,...) with the sources/links.*

Random number generation is a process by which, often by means of a **random number generator** (**RNG**), a sequence of numbers or symbols that cannot be reasonably predicted better than by random chance is generated. This means that the particular outcome sequence will contain some patterns detectable in hindsight but unpredictable to foresight. True random number generators can be *hardware random-number generators* (HRNGS) that generate random numbers, wherein each generation is a function of the current value of a physical environment's attribute that is constantly changing in a manner that is practically impossible to model. This would be in

contrast to so-called "random number generations" done by *pseudorandom number generators* (PRNGs) that generate numbers that only look random but are in fact pre-determined—these generations can be reproduced simply by knowing the state of the PRNG.

Various applications of randomness have led to the development of several different methods for generating random data. Some of these have existed since ancient times, among whose ranks are well-known "classic" examples, including the rolling of dice, coin flipping, the shuffling of playing cards, the use of yarrow stalks (for divination) in the I Ching, as well as countless other techniques. Because of the mechanical nature of these techniques, generating large quantities of sufficiently random numbers (important in statistics) required much work and time. Thus, results would sometimes be collected and distributed as random number tables.

Several computational methods for pseudorandom number generation exist. All fall short of the goal of true randomness, although they may meet, with varying success, some of the statistical tests for randomness intended to measure how unpredictable their results are (that is, to what degree their patterns are discernible). This generally makes them unusable for applications such as cryptography. However, carefully designed *cryptographically secure pseudorandom number generators* (CSPRNGS) also exist, with special features specifically designed for use in cryptography.

Generating a random value with a custom distribution

To generate random values according to a **custom distribution** using a function that generates **uniform random values between 0 and 1** you can use the inverse transform sampling method.

Method Outline

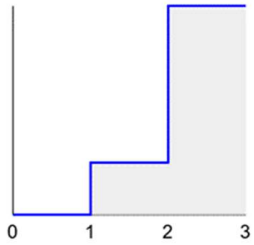
1. First we create a cumulative distribution function (CDF for short)
2. We then mirror the CDF along $y = x$
3. The resulting function can be applied to a random value between 0 and 1

Example

Suppose we want to generate a random point with the following distribution:

- 1/5 of the points uniformly between 1 and 2, and
- 4/5 of the points uniformly between 2 and 3.

The probability density function (PDF) in this case would look like this:

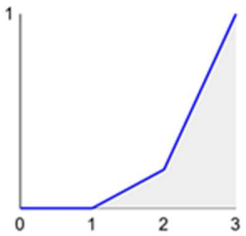


For such simple distribution, inverse transform sampling is a bit of an overkill, but let's go with this since a simple example makes it easier to understand the general idea.

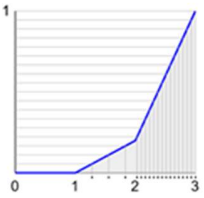
Step 1: Create the CDF

The CDF is, as the name suggests, the cumulative version of the PDF. Intuitively: While PDF(x) describes the number of random values *at* x, CDF(x) describes the number of random values *less than* x.

Since we're working with reals, the CDF is expressed as the integral of the PDF. In this case the CDF would look like:



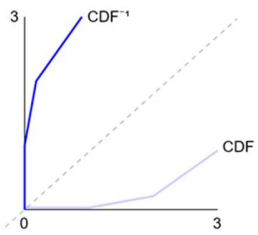
To see why the CDF is useful, imagine that we shoot bullets from left to right at uniformly distributed heights. As the bullets hit the line, they drop down to the ground:



Step 2: Mirror the CDF along $y = x$

The problem is that for this function, the x axis is the *input* and the y axis is the *output*. We can only "shoot bullets from the ground straight up"! We need the inverse function!

This is why we mirror the whole thing; x becomes y and y becomes x:



We call this CDF^{-1}.

Step 3: Apply the resulting function to a uniform value between 0 and 1

With the CDF^{-1} we have all we need. We now simply feed it with uniformly random values between 0 and 1:

$CDF^{-1}(random())$

The result is random values with the desired distribution.

A Real World Example

This technique is used to derive the algorithm for generating a uniformly random point on a disc. See article Generating a random point within a circle (uniformly).

https://programming.guide/generate-random-value-with-distribution.html

Random Number Table

Random Number Generator | Frequently-Asked Questions | Sample Problems

100 Random Numbers

26500 29473 22649 80591 63105 40290 53307 24284 50021 50563 34428 70108 69993 43305 23217 27690 50398
67645 78544 01674 33549 27883 93751 93286 72017 97563 54462 26899 48937 58526 21402 01591 23325 64977
38206 67709 00386 32548 64229 28433 26831 52358 32364 66338 11187 30076 31850 94956 69109 49987 16087
35336 93694 64050 35446 17614 50793 05962 95086 07565 99322 30221 37762 58175 36579 76190 51639 86198
68551 57155 03105 73720 05229 74392 94583 94721 80921 56227 47343 66640 99563 22101 09317 26934 05035
71628 87908 34641 76468 98662 07806 96947 48560 30359 93795 20741 56890 91235 78573 89184

Specs: This table of 100 random numbers was produced according to the following specifications: Numbers were randomly selected from within the range of 0 to 99999. Duplicate numbers were allowed. This table was generated on 12/13/2022.

Print Table

Berman H.B., "*Random Number Generator*", [online] Available at: https://stattrek.com/statistics/random-number-generator URL [Accessed Date: 12/13/2022].

***C) Search on the Internet**: The formulas for calculating the above characteristics and the various coefficients (Pearson, Spearman, Kendal,...)*

$$m = \frac{\text{sum of the terms}}{\text{number of terms}}$$

m = mean

$$\text{Med}(X) = \begin{cases} X[\frac{n+1}{2}] & \text{if n is odd} \\ \frac{X[\frac{n}{2}]+X[\frac{n}{2}+1]}{2} & \text{if n is even} \end{cases}$$

X = ordered list of values in data set

n = number of values in data set

$$\begin{aligned} \sigma^2 &= \frac{\sum(X-\mu)^2}{N} \\ &= \frac{\sum(X^2-2\mu X+\mu^2)}{N} \\ &= \frac{\sum X^2}{N} - \frac{2\mu\sum X}{N} + \frac{N\mu^2}{N} \\ &= \frac{\sum X^2}{N} - 2\mu^2 + \mu^2 \\ &= \frac{\sum X^2}{N} - \mu^2 \end{aligned}$$

$$skewness = \frac{\sum_{i=1}^{N}(x_i-\bar{x})^3}{(N-1)s^3}$$

where:

- σ is the standard deviation
- $\bar{x}$ is the mean of the distribution
- N is the number of observations of the sample

$$kurtosis = \frac{\sum_{i=1}^{N}(x_i-\bar{x})^4}{(N-1)s^4}$$

where:

- σ is the standard deviation
- $\bar{x}$ is the mean of the distribution
- N is the number of observations of the sample

What does Max () mean in math?
The Math.max() function **returns the largest of the numbers given as input parameters, or -Infinity if there are no parameters**.

What is the min function in math?
min() The Math. min() function **returns the smallest of the numbers given as input parameters, or Infinity if there are no parameters**.

How do you calculate Q1 and Q3?
Quartile Formula

1. Quartile Formula (Table of Contents)
2. Let's say we have a data set A which contains 19 data points. ...
3. Data Set:
4. Lower Quartile (Q1) = (N+1) * 1 / 4.
5. Middle Quartile (Q2) = (N+1) * 2 / 4.
6. Upper Quartile (Q3)= (N+1) * 3 / 4.
7. Interquartile Range = Q3 – Q1.
8. Lower Quartile (Q1) = (N+1) * 1 / 4.

Interquartile range (IQR)

In descriptive statistics, the **interquartile range** (**IQR**) is a measure of statistical dispersion, which is the spread of the data. The IQR may also be called the **midspread**, **middle 50%**, **fourth spread**, or **H-spread.** It is defined as the difference between the 75th and 25th percentiles of the data. To calculate the IQR, the data set is divided into quartiles, or four rank-ordered even parts via linear interpolation. These quartiles are denoted by Q_1 (also called the lower quartile), Q_2 (the median),

and Q_3 (also called the upper quartile). The lower quartile corresponds with the 25th percentile and the upper quartile corresponds with the 75th percentile, so IQR = $Q_3 - Q_1$.

The IQR is an example of a trimmed estimator, defined as the 25% trimmed range, which enhances the accuracy of dataset statistics by dropping lower contribution, outlying points. It is also used as a robust measure of scale It can be clearly visualized by the box on a Box plot.

1σ 2σ 3σ formulas

In statistics, the **68–95–99.7 rule**, also known as the **empirical rule**, is a shorthand used to remember the percentage of values that lie within an interval estimate in a normal distribution: 68%, 95%, and 99.7% of the values lie within one, two, and three standard deviations of the mean, respectively.

In mathematical notation, these facts can be expressed as follows, where Pr() is the probability function,[1] X is an observation from a normally distributed random variable, μ (mu) is the mean of the distribution, and σ (sigma) is its standard deviation:

$$\Pr(\mu - 1\sigma \leq X \leq \mu + 1\sigma) \approx 68.27\%$$
$$\Pr(\mu - 2\sigma \leq X \leq \mu + 2\sigma) \approx 95.45\%$$
$$\Pr(\mu - 3\sigma \leq X \leq \mu + 3\sigma) \approx 99.73\%$$

The usefulness of this heuristic especially depends on the question under consideration.

In the empirical sciences, the so-called **three-sigma rule of thumb** (or **3σ rule**) expresses a conventional heuristic that nearly all values are taken to lie within three standard deviations of the mean, and thus it is empirically useful to treat 99.7% probability as near certainty.[2]

In the social sciences, a result may be considered "significant" if its confidence level is of the order of a two-sigma effect (95%), while in particle physics, there is a convention of a five-sigma effect (99.99994% confidence) being required to qualify as a discovery.

A weaker three-sigma rule can be derived from Chebyshev's inequality, stating that even for non-normally distributed variables, at least 88.8% of cases should fall within properly calculated three-sigma intervals. For unimodal distributions, the probability of being within the interval is at least 95% by the Vysochanskij–Petunin inequality. There may be certain assumptions for a distribution that force this probability to be at least 98%.

Covariance

Formula

$$cov_{x,y} = \frac{\sum(x_i - \bar{x})(y_i - \bar{y})}{N - 1}$$

$cov_{x,y}$ = covariance between variable x and y
x_i = data value of x
y_i = data value of y
$\bar{x}$ = mean of x
$\bar{y}$ = mean of y
N = number of data values

In probability theory and statistics, **covariance** is a measure of the joint variability of two random variables.[1] If the greater values of one variable mainly correspond with the greater values of the other variable, and the same holds for the lesser values (that is, the variables tend to show similar behavior), the covariance is positive.[2] In the opposite case, when the greater values of one variable mainly correspond to the lesser values of the other, (that is, the variables tend to show opposite behavior), the covariance is negative. The

sign of the covariance therefore shows the tendency in the linear relationship between the variables. The magnitude of the covariance is not easy to interpret because it is not normalized and hence depends on the magnitudes of the variables. The normalized version of the covariance, the correlation coefficient, however, shows by its magnitude the strength of the linear relation.

A distinction must be made between (1) the covariance of two random variables, which is a population parameter that can be seen as a property of the joint probability distribution, and (2) the sample covariance, which in addition to serving as a descriptor of the sample, also serves as an estimated value of the population parameter.

Properties

One of the key properties of the covariance is the fact that **independent random variables have zero covariance**. Covariance of independent variables. If X X X and Y Y Y are independent random variables, then Cov (X , Y) = 0.

Purpose

Covariance is a statistical tool that is used **to determine the relationship between the movements of two random variables**. When two stocks tend to move together, they are seen as having a positive covariance; when they move inversely, the covariance is negative.

Differences

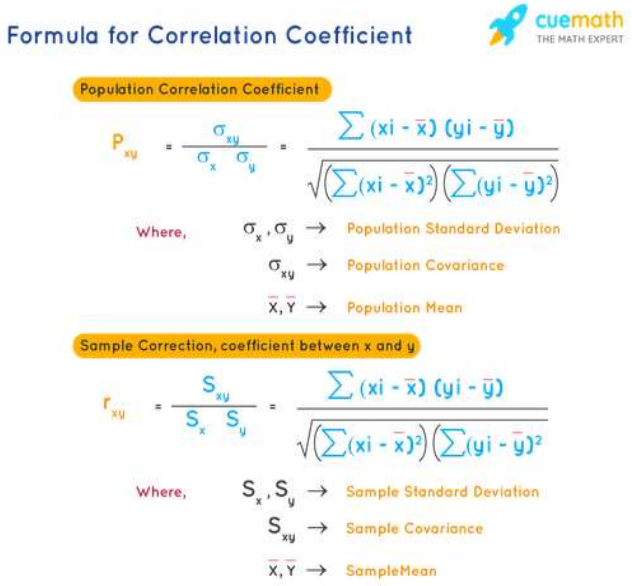
Covariance shows you how the two variables differ, whereas correlation shows you how the two variables are related

Principles

'Covariation principle' was introduced by Harold Kelley who defined it as **attribution of an effect to one of its possible causes with which it covaries over a period time**. Covariation principle applies to the situations in which the attributors observed or noticed the effect two or more times.

What Is the Correlation Coefficient Formula?

The correlation coefficient is a statistical concept. It establishes a relation between predicted and actual values obtained at the end of a statistical experiment. The correlation coefficient formula helps to calculate the relationship between two variables and thus the result so obtained explains the exactness between the predicted and actual values.



Pearson correlation coefficient (PCC)

$$r = \frac{\sum (x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum (x_i - \bar{x})^2 \sum (y_i - \bar{y})^2}}$$

r = correlation coefficient
x_i = values of the x-variable in a sample
$\bar{x}$ = mean of the values of the x-variable
y_i = values of the y-variable in a sample
$\bar{y}$ = mean of the values of the y-variable

In statistics, the **Pearson correlation coefficient** (**PCC**) — also known as **Pearson's r**, the **Pearson product-moment correlation coefficient** (**PPMCC**), the **bivariate correlation**,[1] or colloquially simply as **the correlation coefficient**[2] — is a measure of linear correlation between two sets of data. It is the ratio between the covariance of two variables and the product of their standard deviations; thus, it is essentially a normalized measurement of the covariance, such that the result always has a value between −1 and 1. As with covariance itself, the measure can only reflect a linear correlation of variables, and ignores many other types of relationships or correlations. As a simple example, one would expect the age and height of a sample of teenagers from a high school to have a Pearson correlation coefficient significantly greater than 0, but less than 1 (as 1 would represent an unrealistically perfect correlation).

Range

+ 1 through 0 to – 1

The correlation coefficient is measured on a scale that varies from **+ 1 through 0 to – 1**. Complete correlation between two variables is expressed by either + 1 or -1. When one variable increases as the other increases the correlation is positive; when one decreases as the other increases it is negative.

Results

The Pearson correlation measures the strength of the linear relationship between two variables. It has a value between -1 to 1, with a value of **-1 meaning a total negative linear correlation, 0 being no correlation, and + 1 meaning a total positive correlation**.

Determination

What is the Coefficient of Determination? **The coefficient of determination, r^2, is the square of the Pearson correlation coefficient r (i.e., r^2)**. So, for example, a Pearson correlation coefficient of 0.6 would result in a coefficient of determination of 0.36, (i.e., $r^2 = 0.6 \times 0.6 = 0.36$).

Data

The Pearson correlation coefficient (r) is the most common way of measuring a linear correlation. It is **a number between –1 and 1 that measures the strength and direction of the relationship between two variables**. When one variable changes, the other variable changes in the same direction.

Spearman's rank correlation coefficient

$$\rho = 1 - \frac{6 \sum d_i^2}{n(n^2 - 1)}$$

ρ = Spearman's rank correlation coefficient

d_i = difference between the two ranks of each observation

n = number of observations

In statistics, **Spearman's rank correlation coefficient** or **Spearman's *ρ***, named after Charles Spearman and often denoted by the Greek letter **ϱ** (rho) or as r_s, is a nonparametric measure of rank correlation (statistical dependence between the rankings of two variables). It assesses how well the relationship between two variables can be described using a monotonic function.

The Spearman correlation between two variables is equal to the Pearson correlation between the rank values of those two variables; while Pearson's correlation assesses linear relationships, Spearman's correlation assesses monotonic relationships (whether linear or not). If there are no repeated data values, a perfect Spearman correlation of +1 or −1 occurs when each of the variables is a perfect monotone function of the other.

Intuitively, the Spearman correlation between two variables will be high when observations have a similar (or identical for a correlation of 1) rank (i.e. relative position label of the observations within the variable: 1st, 2nd, 3rd, etc.) between the two variables, and low when observations have a dissimilar (or fully opposed for a correlation of −1) rank between the two variables.

Significance

If you set α = 0.05, achieving a statistically significant Spearman rank-order correlation means that you can be sure that there is less than a 5% chance that the strength of the relationship you found (your ρ coefficient) happened by chance if the null hypothesis were true.

Differences

The fundamental difference between the two correlation coefficients is that **the Pearson coefficient works with a linear relationship between the two variables whereas the Spearman Coefficient works with monotonic relationships as well**.

Data

Spearman's correlation works by **calculating Pearson's correlation on the ranked values of this data**. Ranking (from low to high) is obtained by assigning a rank of 1 to the lowest value, 2 to the next lowest and so on. If we look at the plot of the ranked data, then we see that they are perfectly linearly related.

Range

Spearman's correlation ranges in value from **-1 to 1**, with values near 1 indicating similarity in ranks for the two variables and values near -1 indicating ranks are dissimilar for the two variables.

Kendall rank correlation coefficient

In statistics, the **Kendall rank correlation coefficient**, commonly referred to as **Kendall's τ coefficient** (after the Greek letter τ, tau), is a statistic used to measure the ordinal association

between two measured quantities. A **τ test** is a non-parametric hypothesis test for statistical dependence based on the τ coefficient.

It is a measure of rank correlation: the similarity of the orderings of the data when ranked by each of the quantities. It is named after Maurice Kendall, who developed it in 1938,[1] though Gustav Fechner had proposed a similar measure in the context of time series in 1897.[2]

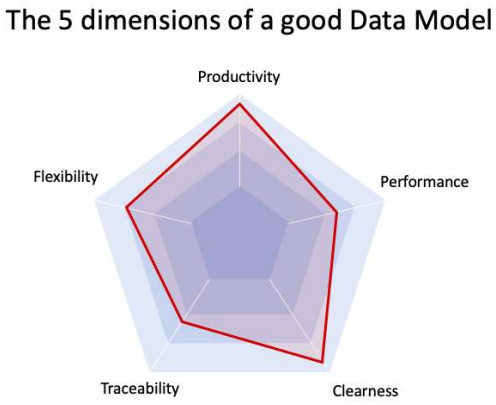
Intuitively, the Kendall correlation between two variables will be high when observations have a similar (or identical for a correlation of 1) rank (i.e. relative position label of the observations within the variable: 1st, 2nd, 3rd, etc.) between the two variables, and low when observations have a dissimilar (or fully different for a correlation of −1) rank between the two variables.

3. KEY STATISTICAL CHARACTERISTICS OF THE DATA - IMPORTANCE OF IMPRESSION

A) Search the Internet*: Online tools with the basic types of statistical attribute calculation for your data*
https://statpages.info/

B) You search the Internet*: Advanced ways of representing the data to help model it.*

How Can We Define a Good Data Model?
In other words, how can we compare various Data Modeling options? What factors should be taken into account?. I believe the following five dimensions are the most important:



1) Performance
This is a vast topic, and we are not discussing database vendors, data indexing, or technical modifications to boost read and write speeds. I believe we can ascribe performance advantages solely based on how we model the data.

2) Productivity
On the developer side, we want a model that is simple to work with and reason about, so we can "create a lot of good code" without wasting time (the concept of productivity).

3) Clearness
The Data Model's ability to be comprehended by those who look at it. As you may have heard, most developers read code rather than write it, therefore we must clearly grasp what we are doing with our data.

4) Flexibility
The Model's capacity to evolve without having a significant influence on our code. Because the startup you work for is evolving, the systems and Data Models that power it will need to evolve as well.

5) Traceability

Finally, we want to have data that is useful to the system as well as data that is valuable to our users. Knowing what happened in the past, what values the entities had at some point in time, being able to travel back and forth in time, and so on.

What is Advanced-Data Modeling and its Techniques?

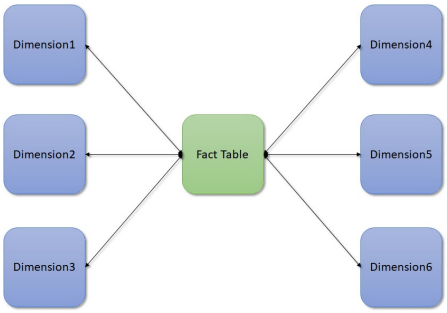


Image Source

The Application Designer can use Advanced Modeling to include items that are not automatically included in the program. This can aid Comprehension of the Application's Views, Dimensions, and Approval Hierarchy.

When the program is deployed to the distributed client, the application designer can verify that those objects are included inside the scope of the application.

The following are five different types of data organization techniques:

1) Advanced-Data Modeling Concepts: Technique of Hierarchy

A Tree-like structure characterizes the hierarchical model. There is one root node or one parent node, and the other child nodes are arranged in a specific order. However, the hierarchical approach is currently rarely employed. This paradigm is applicable to real-world model relationships.

2) Advanced-Data Modeling Concepts: Object-Oriented Model

The Object-Oriented Method involves the construction of objects that contain values that have been saved. The Object-Oriented Model allows for communication while also allowing for Data Abstraction, Inheritance, and Encapsulation.

3) Advanced-Data Modeling Concepts: Networking Methodology

The network model allows us to represent items and their relationships in a flexible manner. It has a feature called a schema that represents the data as a graph. An object is represented within a node, and the relationship between them is represented as an edge, allowing them to keep many parent and child records in a generalized fashion.

4) Advanced-Data Modeling Concepts: Entity-Relationship Diagram

The ER model (Entity-Relationship model) is a high-level relational model used to specify data pieces and relationships for system entities. This conceptual design provides a clearer view of the data, making it easier for us to interpret. The complete database is represented in this paradigm by an entity-relationship diagram, which is made up of Entities, Attributes, and Relationships.

5) Advanced-Data Modeling Concepts: Relational Methodology

The term "relational" refers to the various relationships that exist between the entities. There are also many sets of relationships between the entities, such as one to one, one to many, many to one, and many to many.

Benefits of Advanced-Data Modeling Concepts

Advanced-Data Modeling Concepts also ensure that particular Databases and Apps include the correct data and are built to suit business data processing and management requirements.

Other advantages of Advanced-Data Modeling include the following:

- **Internal Consensus on Data Definition is required:** Data Modeling aids attempt to standardize data definitions, language, concepts, and formats across the company.
- **Business Users are Involved in Data Management:** Data Modeling necessitates business input, it fosters collaboration among data management teams and business stakeholders, resulting in better systems.
- **Database Design that is More Efficient and Less Expensive:** Data Modeling accelerates database designers' work and decreases the chance of design errors that necessitate adjustments later in the process by providing them with a precise blueprint to work from.
- **Improved Utilization of Accessible Data Assets:** Finally, Efficient Data Modeling enables firms to make better use of their data, which can lead to improved company performance, new business prospects, and a competitive advantage over competitors.

Limitations of Advanced-Data Modeling Concepts

However, Advanced-Data Modeling is a complex process that can be tough to master. These are some of the most prevalent issues that might derail Data Modeling projects:

- **Inadequate Organisational Commitment:** It's difficult to acquire the necessary level of company participation if corporate and business executives aren't on board with the need for Data Modeling. As a result, data management teams must obtain executive support from the start.
- **Lack of Comprehension of Business Users:** Even when business stakeholders are completely committed, Data Modeling is an abstract process that can be difficult for individuals to understand. To avoid this, conceptual and logical data models should be built around business terms and concepts.
- **The Complexity of Modeling:** Advanced-Data Modeling Concepts are frequently large and complicated, and modeling efforts can become cumbersome if teams keep creating new versions without finalizing the designs. It is critical to establish priorities and commit to a project scope that is manageable.
- **Undefined or Ambiguous Business Need:** The business side may not have completely developed information demands, especially with new applications. Data Modelers frequently need to ask a series of questions in order to acquire or clarify requirements and find the relevant data.

Conclusion

In a nutshell, Advanced-Data Modeling Concepts aids in Data Visualisation. Data Models are created during the project's design and analysis phases to guarantee that the application requirements are met. This is what Advanced-Data Modeling Technique provides for us.

Hevo Data is a no-code data pipeline that can instantly connect multiple sources. Integrating and analyzing data from a large number of disparate sources can be difficult; this is where Hevo comes in.

C) Report to the Internet*: The following data files:*

https://en.wikipedia.org/wiki/Iris_flower_data_set - Περιγραφή (*Wikipedia)*

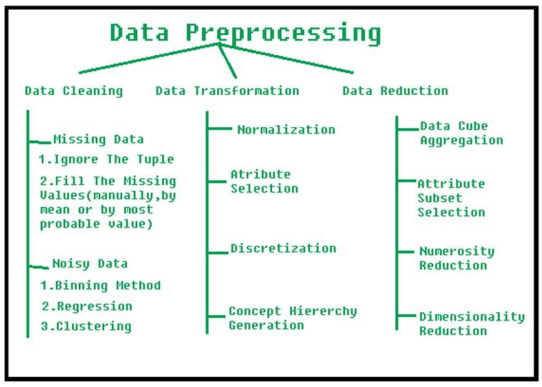
https://archive.ics.uci.edu/ml/datasets/Iris - Download *από το Data Folder*

4. DATA PRE-PROCESSING STAGES

What are five forms of data preprocessing?

Important Data Preprocessing Techniques

- Data Cleaning.
- Dimensionality Reduction.
- Feature Engineering.
- Sampling Data.
- Data Transformation.
- Imbalanced Data.



Steps Involved in Data Preprocessing:

1. Data Cleaning:
The data can have many irrelevant and missing parts. To handle this part, data cleaning is done. It involves handling of missing data, noisy data etc.

- **(a). Missing Data:**

This situation arises when some data is missing in the data. It can be handled in various ways.
Some of them are:

1. **Ignore the tuples:**
   This approach is suitable only when the dataset we have is quite large and multiple values are missing within a tuple.
2. **Fill the Missing values:**
   There are various ways to do this task. You can choose to fill the missing values manually, by attribute mean or the most probable value.

- **(b). Noisy Data:**
  Noisy data is a meaningless data that can't be interpreted by machines.It can be generated due to faulty data collection, data entry errors etc. It can be handled in following ways :
  1. **Binning Method:**
     This method works on sorted data in order to smooth it. The whole data is divided into segments of equal size and then various methods are performed to complete the task. Each segmented is handled separately. One can replace all data in a segment by its mean or boundary values can be used to complete the task.
  2. **Regression:**
     Here data can be made smooth by fitting it to a regression function.The regression used may be linear (having one independent variable) or multiple (having multiple independent variables).
  3. **Clustering:**
     This approach groups the similar data in a cluster. The outliers may be undetected or it will fall outside the clusters.

2. Data Transformation:
This step is taken in order to transform the data in appropriate forms suitable for mining process. This involves following ways:

1. **Normalization:**
   It is done in order to scale the data values in a specified range (-1.0 to 1.0 or 0.0 to 1.0)
2. **Attribute Selection:**
   In this strategy, new attributes are constructed from the given set of attributes to help the mining process.
3. **Discretization:**
   This is done to replace the raw values of numeric attribute by interval levels or conceptual levels.
4. **Concept Hierarchy Generation:**
   Here attributes are converted from lower level to higher level in hierarchy. For Example- The attribute “city” can be converted to “country”.

3. Data Reduction:
Since data mining is a technique that is used to handle huge amount of data. While working with huge volume of data, analysis became harder in such cases. In order to get rid of this, we uses data reduction technique. It aims to increase the storage efficiency and reduce data storage and analysis costs.

The various steps to data reduction are:

1. **Data Cube Aggregation:**
   Aggregation operation is applied to data for the construction of the data cube.
2. **Attribute Subset Selection:**
   The highly relevant attributes should be used, rest all can be discarded. For performing attribute selection, one can use level of significance and p- value of the attribute.the attribute having p-value greater than significance level can be discarded.
3. **Numerosity Reduction:**
   This enable to store the model of data instead of whole data, for example: Regression Models.
4. **Dimensionality Reduction:**
   This reduce the size of data by encoding mechanisms.It can be lossy or lossless. If after reconstruction from compressed data, original data can be retrieved, such reduction are called lossless reduction else it is called lossy reduction. The two effective methods of dimensionality reduction are:Wavelet transforms and PCA (Principal Component Analysis).

5. APPLICATIONS OF FUZZY LOGIC IN SPECIALTY

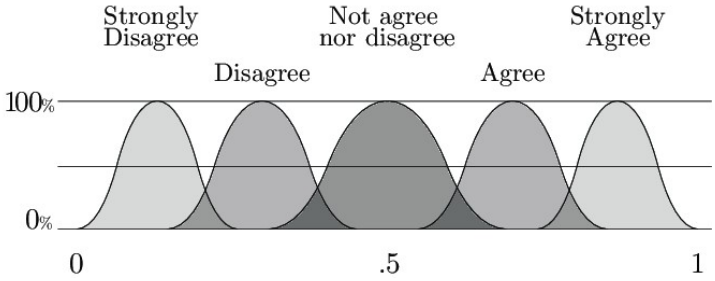
Search the Internet: *Applications of Fuzzy Sets in your specialty or in everyday life, and present their basic rules schematically, as in the image above.*

What is the application of fuzzy sets?

Fuzzy logic has been successfully used in numerous fields such as **control systems engineering, image processing, power engineering, industrial automation, robotics, consumer electronics, and optimization**. This branch of mathematics has instilled new life into scientific fields that have been dormant for a long time.

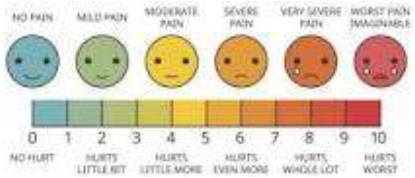
How fuzzy logic is used in NLP?

Fuzzy Logic is a type of Natural Language Processing (NLP) that **helps identify and group similar business records**. It tries to associate similar business records that were misrepresented or misspelled. Hence, we will obtain a cleansed dataset.



What is a Likert scale explain?

A Likert scale is **a type of rating scale, often found on survey forms or questionnaires, that measures how people feel about something which can be useful in many different situations.**



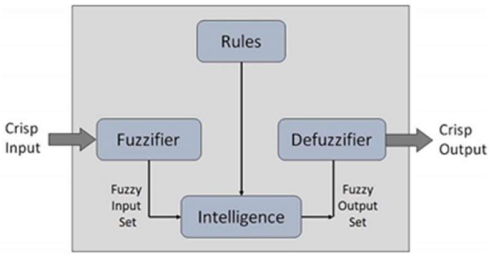
What are the applications of fuzzy logic controller?

The fuzzy logic controller can be used to automatic control systems, for example, autonomous vehicle systems. Also we can use the concept of Fuzzy Set Theory to help us make decision in various areas such as in finance and management.

Which of the following are the application areas of fuzzy logic?

Pattern Recognition and Classification

- Fuzzy logic based speech recognition.
- Fuzzy logic based.
- Handwriting recognition.
- Fuzzy logic based facial characteristic analysis.
- Command analysis.
- Fuzzy image search.



What are the real life applications of fuzzy systems explain in details?

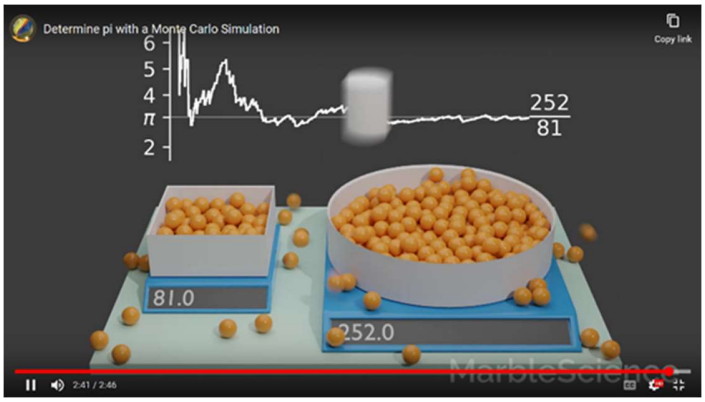
The Fuzzy Logic can be used in a variety of industries, including domestic goods, automotive systems, environment control, etc. Some of them are: **It is used to control the altitude of aircraft, satellites, and spaceships**.

What are the different types of fuzzy sets?

Index Terms—Type-2 fuzzy set; Set-valued fuzzy set; Hesitant fuzzy set; Interval-valued fuzzy set; Atanassov intuitionistic fuzzy set; Interval type-2 fuzzy sets; Interval-valued Atanassov intuitionistic fuzzy set; Neutrosophic set; Bipolar-valued fuzzy set; Fuzzy multiset; Fuzzy rough set; Fuzzy soft set; Multi-polar-valued fuzzy set.

6. MODEL SIMULATION AND FINDING P. SEARCH APPLICATIONS OF MONTE-CARLO IN SPECIALTY

***A) Search the Internet**: The link above and observe the variation of the result as the experiment progresses.*



***B) Search the Internet**: Applications of Monte-Carlo in your specialty. List the sources/links.*

What is Monte Carlo model of simulation?

Definition: Monte Carlo Simulation is **a mathematical technique that generates random variables for modelling risk or uncertainty of a certain system**. The random variables or inputs are modelled on the basis of probability distributions such as normal, log normal, etc.

What is the difference between simulation and Monte Carlo simulation?

Sawilowsky distinguishes between a simulation, a Monte Carlo method, and a Monte Carlo simulation: a simulation is a fictitious representation of reality, a Monte Carlo method is a technique that can be used to solve a mathematical or statistical problem, and a Monte Carlo simulation uses repeated sampling to obtain ...

Applied statistics

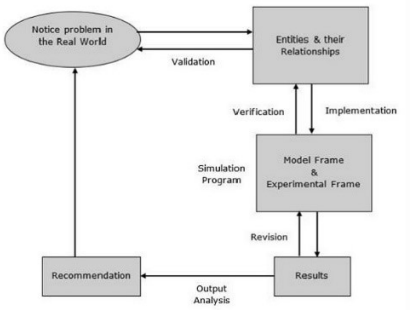
The standards for Monte Carlo experiments in statistics were set by Sawilowsky. In applied statistics, Monte Carlo methods may be used for at least four purposes:

1. To compare competing statistics for small samples under realistic data conditions. Although type I error and power properties of statistics can be calculated for data drawn from classical theoretical distributions (*e.g.*, normal curve, Cauchy distribution) for

asymptotic conditions (*i. e*, infinite sample size and infinitesimally small treatment effect), real data often do not have such distributions.

2. To provide implementations of hypothesis tests that are more efficient than exact tests such as permutation tests (which are often impossible to compute) while being more accurate than critical values for asymptotic distributions.
3. To provide a random sample from the posterior distribution in Bayesian inference. This sample then approximates and summarizes all the essential features of the posterior.
4. To provide efficient random estimates of the Hessian matrix of the negative log-likelihood function that may be averaged to form an estimate of the Fisher information matrix.

Monte Carlo methods are also a compromise between approximate randomization and permutation tests. An approximate randomization test is based on a specified subset of all permutations (which entails potentially enormous housekeeping of which permutations have been considered). The Monte Carlo approach is based on a specified number of randomly drawn permutations (exchanging a minor loss in precision if a permutation is drawn twice—or more frequently—for the efficiency of not having to track which permutations have already been selected).



What are the 5 applications of modeling and simulation?

Modelling & Simulation can be applied to the following areas – **Military applications, training & support, designing semiconductors, telecommunications, civil engineering designs & presentations, and E-business models**.

Monte Carlo simulation applications in business

Monte Carlo simulation is useful for a wide range of challenges in business, such as the relatively simple determination of probable product demand or the calculation of complex business risks. These applications of Monte Carlo simulation are possible due to developments in modern computation. As companies gain greater access to more powerful multi-core processors and cloud computing, the challenges that can be met with simulation and Monte Carlo optimization will continue to expand.

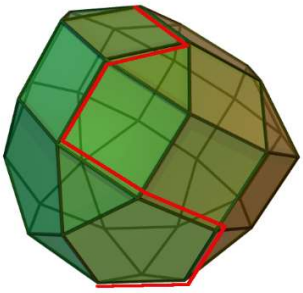
AnyLogic enables Monte Carlo simulation for highly complex systems. With multimethod modeling, simulated systems can be complex, dynamic, and non-linear. The results from these simulation models can come from parallel processing and cloud computing and made available in a variety of ways, including via API and custom UI.

https://www.anylogic.com/blog/monte-carlo-simulation-in-business/

7. SIMPLE AND COMPLEX METHODS OF SEARCHING FOR THE OPTIMUM THAT WE HAVE ALREADY TAUGHT

***A) Search the Internet**: Optimization Methods you have been taught and give a brief description with an example application.*

Optimization: Simplex Method for Maximization.



Introduction

The Simplex method is an approach to solving linear programming models by hand using slack variables, tableaus, and pivot variables as a means to finding the optimal solution of an optimization problem. A linear program is a method of achieving the best outcome given a maximum or minimum equation with linear constraints. Most linear programs can be solved using an online solver such as MatLab, but the Simplex method is a technique for solving linear programs by hand. To solve a linear programming model using the Simplex method the following steps are necessary:

- Standard form
- Introducing slack variables
- Creating the tableau
- Pivot variables
- Creating a new tableau
- Checking for optimality
- Identify optimal values

This document breaks down the Simplex method into the above steps and follows the example linear programming model shown below throughout the entire document to find the optimal solution.

Maximize: $p = 4x + 5y + 6z$

Which is subject to three constraints:

$$\text{Subject to} \begin{cases} 2x + 3y + z \leq 900 \\ 3x + y + z \leq 350 \\ 4x + 2y + z \leq 400 \\ x \geq 0,\ y \geq 0,\ z \geq 0 \end{cases}$$

Maximize:

$$p = 4x + 5y + 6z$$

To Identify Maximize we must follow below steps.

Steps:

1. Convert all inequality subject equations by adding slack variables. Also create objective function equal to zero by moving all values on one side. Equations after adding slack variable (u,v,w) Where u,v,w≥0

$$\begin{cases} 2x + 3y + z + u = 900 \\ 3x + y + z + v = 350 \\ 4x + 2y + z + w = 400 \\ -4x - 5y - 6z + p = 0 \\ x \geq 0,\ y \geq 0,\ z \geq 0 \end{cases}$$

2. A Simplex tableau is used to perform row operations on the linear programming model as well as to check a solution for optimality. The tableau consists of the coefficient corresponding to the linear constraint variables and the coefficients of the objective function. In the tableau below, the bolded top row of the tableau states what each column represents. The following two rows represent the linear constraint variable coefficients from the linear programming model, and the last row represents the objective function variable coefficients.

Now create simplex tableau by creating coefficients of all subject equation and add Object equation at the bottom.

x	y	z	u	v	w	p	Constant
2	3	1	1	0	0	0	900
3	1	1	0	1	0	0	350
4	1	1	0	0	1	0	400
-4	-5	-6	0	0	0	1	0

3. To calculate ratio, we can select the minimum value from the last row and follow below steps.

x	y	z	u	v	w	p	Constant
2	3	1	1	0	0	0	900
3	1	1	0	1	0	0	350
4	1	1	0	0	1	0	400
-4	-5	-6	0	0	0	1	0

How to identify Pivot column:

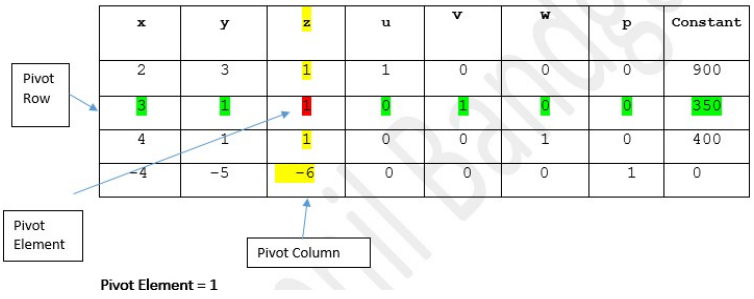
In the above table, **-6** is the smallest negative in the last row. This will designate the **z** column to contain the pivot variable which is highlighted by yellow.

How to identify Pivot row:

The pivot variable is used in row operations to identify which variable will become the unit value and is a key factor in the conversion of the unit value. The pivot variable can be identified by looking at the bottom row of the tableau and the indicator. Assuming that the solution is not optimal, pick the smallest negative value in the bottom row. One of the values lying in the column of this value will be the pivot variable. To find the indicator, divide the beta values of the linear constraints by their corresponding values from the column containing the possible pivot variable. The intersection of the row with the smallest non-negative indicator and the smallest negative value in the bottom row will become the pivot variable.

Divide the pivot column with constant in corresponding row and identify the values.

Solving for the ratio gives us a value of **(900/1 = 900)** for the first constraint, a value of **(350/1 = 350)** for the second constraint, and value of third constraint **(400/1 = 400)**. Due to 350 being the smallest non-negative ratio, the pivot value will be in the second row which is highlighted in green and intersection for pivot row and column will be pivot element which has a value of 1 which is highlighted in Red color.



Now that the pivot variable has been identified, we can work on further solution to make it optimize.

4. To optimize the pivot variable, it will need to be transformed into a unit value (value of 1). As, the pivot element is already 1 here we do not have to make it unit value.

x	y	z	u	v	w	p	Constant
2	3	1	1	0	0	0	900
3	1	1	0	1	0	0	350
4	1	1	0	0	1	0	400
-4	-5	-6	0	0	0	1	0

5. After the unit value has been determined, Work on formula which will make the other values in the column containing the unit value will become zero. This is because slack variable and other variable value can be identified, and solution is being optimized.

Formulas: R1=R1-R2 , R3=R3-R2 and R4=R4+6R2

Applying above formulas on our simplex tableau will result into below table.

x	y	z	u	v	w	p	Constant
-1	2	0	1	-1	0	0	550
3	1	1	0	1	0	0	350
1	0	0	0	-1	1	0	50
14	1	0	0	6	0	1	2100

The optimal solution is obtained because all values in the bottom row are greater than or equal to zero.

Here, basic variables are u, z, w, and p.
Non basic variables are x, v and y.

From this we can obtain variable values as below,

x=0, y=0, v=0, u=550, w = 50, p = 2100, z = 350.

Now to validate the equation,

$$p = 4x + 5y + 6z$$
$$p = 4(0) + 5(0) + 6(350)$$
$$p = 2100$$

The maximum optimal value is 2100 and found at (0,0, 350) of the objective function.

Conclusion

The Simplex method is an approach for determining the optimal value of a linear program by hand. The method produces an optimal solution to satisfy the given constraints and produce a maximum zeta value. To use the Simplex method, a given linear programming model needs to be in standard

form, where slack variables can then be introduced. Using the tableau and pivot variables, an optimal solution can be reached.

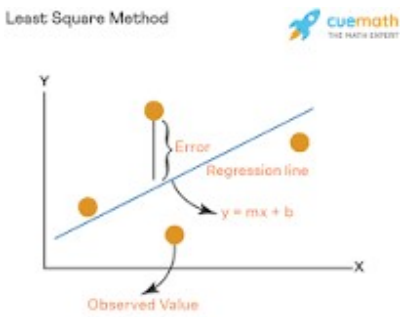
https://medium.com/analytics-vidhya/optimization-simplex-method-for-maximization-e117dfa38114

least squares

What is least squares optimization?

Least squares (LS) optimization problems are those in which the objective (error) function is a quadratic function of the parameter(s) being optimized.

What is least square method with example?



Example: Let's say we have data as shown below. Solution: We will follow the steps to find the linear line. So, the required equation of least squares is y = mx + b = 13/10x + 5.5/5. The least-squares method is **used to predict the behavior of the dependent variable with respect to the independent variable**.

Generalized least squares

In statistics, generalized least squares (GLS) is a technique for estimating the unknown parameters in a linear regression model when there is a certain degree of correlation between the residuals in a regression model. In these cases, ordinary least squares and weighted least squares can be statistically inefficient, or even give misleading inferences. GLS was first described by Alexander Aitken in 1936.

https://homepage.ntu.edu.tw/~ckuan/pdf/et01/et_Ch4.pdf

https://www.cns.nyu.edu/~eero/math-tools/Handouts/leastSquares.pdf

Non-linear least squares

Non-linear least squares is the form of least squares analysis used to fit a set of *m* observations with a model that is non-linear in *n* unknown parameters ($m \geq n$). It is used in some forms of nonlinear regression. The basis of the method is to approximate the model by a linear one and to refine the parameters by successive iterations. There are many similarities to linear least squares, but also some significant differences. In economic theory, the non-linear least squares method is applied in (i) the probit regression, (ii) threshold regression, (iii) smooth regression, (iv) logistic link regression, (v) Box-Cox transformed regressors ($m(x, \theta_i) = \theta_1 + \theta_2 x^{(\theta_3)}$).

Example: **Curve fitting**

Curve fitting[1][2] is the process of constructing a curve, or mathematical function, that has the best fit to a series of data points,[3] possibly subject to constraints.[4][5] Curve fitting can involve either interpolation ,[6][7] where an exact fit to the data is required, or smoothing,[8][9] in which a

"smooth" function is constructed that approximately fits the data. A related topic is regression analysis,[10][11] which focuses more on questions of statistical inference such as how much uncertainty is present in a curve that is fit to data observed with random errors. Fitted curves can be used as an aid for data visualization,[12][13] to infer values of a function where no data are available,[14] and to summarize the relationships among two or more variables.[15] Extrapolation refers to the use of a fitted curve beyond the range of the observed data,[16] and is subject to a degree of uncertainty[17] since it may reflect the method used to construct the curve as much as it reflects the observed data.

For linear-algebraic analysis of data, "fitting" usually means trying to find the curve that minimizes the vertical (*y*-axis) displacement of a point from the curve (e.g., ordinary least squares). However, for graphical and image applications, geometric fitting seeks to provide the best visual fit; which usually means trying to minimize the orthogonal distance to the curve (e.g., total least squares), or to otherwise include both axes of displacement of a point from the curve. Geometric fits are not popular because they usually require non-linear and/or iterative calculations, although they have the advantage of a more aesthetic and geometrically accurate result.

https://en.wikipedia.org/wiki/Curve_fitting#/media/File:Regression_pic_assymetrique.gif

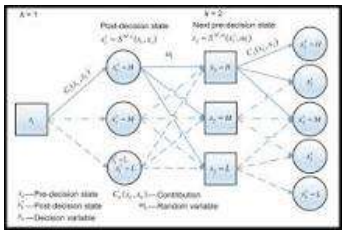
Dynamic programming

Sukanta Nayak, in Fundamentals of Optimization Techniques with Algorithms, 2020

Dynamic programming (DP) is an algorithmic approach for investigating an optimization problem by splitting into several simpler subproblems. It is noted that the overall problem depends on the optimal solution to its subproblems. Hence, the very essential feature of DP is the proper structuring of optimization problems into multiple levels, which are solved sequentially one level at a time. By using ordinary optimization problem techniques, each one level is solved, and its solution helps to define the characteristics of the next level problem in the sequence. Commonly, the levels represent different time periods in the outlook of the overall problem. Understand the basic concept of DP by using a very basic example of the Fibonacci series.

Example 7.1

Fibonacci series is a series of number, which starts with 0 and 1, where each number is the sum of the previous two numbers. The first few Fibonacci numbers are 0, 1, 1, 2, 3, 5, 8, 13, and 21. Now, if it asked to calculate *n*th Fibonacci Number, then we may calculate it by the following generalized formula:



(7.1)Fib(n)=Fib(n−1)+Fib(n−2),forn>1.

Multi-modal or global optimization

Optimization problems are often multi-modal; that is, they possess multiple good solutions. They could all be globally good (same cost function value) or there could be a mix of globally good and locally good solutions. Obtaining all (or at least some of) the multiple solutions is the goal of a multi-modal optimizer.

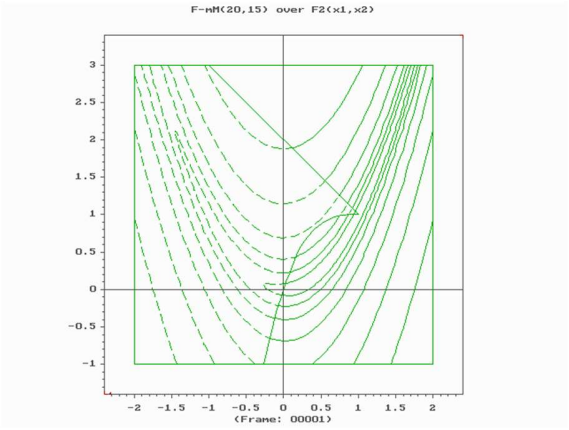
Classical optimization techniques due to their iterative approach do not perform satisfactorily when they are used to obtain multiple solutions, since it is not guaranteed that different solutions will be obtained even with different starting points in multiple runs of the algorithm.

Common approaches to global optimization problems, where multiple local extrema may be present include evolutionary algorithms, Bayesian optimization and simulated annealing.

Evolutionary algorithm

In computational intelligence (CI), an **evolutionary algorithm** (**EA**) is a subset of evolutionary computation,[1] a generic population-based metaheuristic optimization algorithm. An EA uses mechanisms inspired by biological evolution, such as reproduction, mutation, recombination, and selection. Candidate solutions to the optimization problem play the role of individuals in a population, and the fitness function determines the quality of the solutions (see also loss function). Evolution of the population then takes place after the repeated application of the above operators.

Evolutionary algorithms often perform well approximating solutions to all types of problems because they ideally do not make any assumption about the underlying fitness landscape. Techniques from evolutionary algorithms applied to the modeling of biological evolution are generally limited to explorations of microevolutionary processes and planning models based upon cellular processes. In most real applications of EAs, computational complexity is a prohibiting factor.[2] In fact, this computational complexity is due to fitness function evaluation. Fitness approximation is one of the solutions to overcome this difficulty. However, seemingly simple EA can solve often complex problems;[3][4][5] therefore, there may be no direct link between algorithm complexity and problem complexity.



A two-population EA search over a constrained Rosenbrock function with bounded global optimum .

https://en.wikipedia.org/wiki/Evolutionary_algorithm#/media/File:Two-population_EA_search_(2).gif

Bayesian optimization

Bayesian optimization is a sequential design strategy for global optimization of black-box functions[1][2][3] that does not assume any functional forms. It is usually employed to optimize expensive-to-evaluate functions.

Simulated annealing

Simulated annealing (**SA**) is a probabilistic technique for approximating the global optimum of a given function. Specifically, it is a metaheuristic to approximate global optimization in a large search space for an optimization problem. It is often used when the search space is discrete (for example the traveling salesman problem, the boolean satisfiability problem, protein structure

prediction, and job-shop scheduling). For problems where finding an approximate global optimum is more important than finding a precise local optimum in a fixed amount of time, simulated annealing may be preferable to exact algorithms such as gradient descent or branch and bound.

The name of the algorithm comes from annealing in metallurgy, a technique involving heating and controlled cooling of a material to alter its physical properties. Both are attributes of the material that depend on their thermodynamic free energy. Heating and cooling the material affects both the temperature and the thermodynamic free energy or Gibbs energy. Simulated annealing can be used for very hard computational optimization problems where exact algorithms fail; even though it usually achieves an approximate solution to the global minimum, it could be enough for many practical problems.

The problems solved by SA are currently formulated by an objective function of many variables, subject to several constraints. In practice, the constraint can be penalized as part of the objective function.

Similar techniques have been independently introduced on several occasions, including Pincus (1970), Khachaturyan et al (1979, 1981), Kirkpatrick, Gelatt and Vecchi (1983), and Cerny (1985). In 1983, this approach was used by Kirkpatrick, Gelatt Jr., Vecchi,[5] for a solution of the traveling salesman problem. They also proposed its current name, simulated annealing.

This notion of slow cooling implemented in the simulated annealing algorithm is interpreted as a slow decrease in the probability of accepting worse solutions as the solution space is explored. Accepting worse solutions allows for a more extensive search for the global optimal solution. In general, simulated annealing algorithms work as follows. The temperature progressively decreases from an initial positive value to zero. At each time step, the algorithm randomly selects a solution close to the current one, measures its quality, and moves to it according to the temperature-dependent probabilities of selecting better or worse solutions, which during the search respectively remain at 1 (or positive) and decrease toward zero.

The simulation can be performed either by a solution of kinetic equations for density functions or by using the stochastic sampling method. The method is an adaptation of the Metropolis–Hastings algorithm, a Monte Carlo method to generate sample states of a thermodynamic system, published by N. Metropolis et al. in 1953.

https://en.wikipedia.org/wiki/Simulated_annealing#/media/File:Travelling_salesman_problem_solved_with_simulated_annealing.gif

https://en.wikipedia.org/wiki/Simulated_annealing#/media/File:3D_TSP_solved_with_simulated_annealing_2.5_MB.gif

8. TO RECORD VARIOUS KNOWN SEARCH PROBLEMS AS WELL AS VARIOUS ALGORITHM TESTING FUNCTIONS

A) Search the Internet*: Difficult to search/solve problems for which fast methods of solving/finding have not yet been found.*

NP-complete problem, any of a class of computational problems for which no efficient solution algorithm has been found. Many significant computer-science problems belong to this class—e.g., the traveling salesman problem, satisfiability problems, and graph-covering problems.

So-called easy, or tractable, problems can be solved by computer algorithms that run in polynomial time; i.e., for a problem of size *n*, the time or number of steps needed to find the solution is a polynomial function of *n*. Algorithms for solving hard, or intractable, problems, on the other hand, require times that are exponential functions of the problem size *n*. Polynomial-time algorithms are considered to be efficient, while exponential-time algorithms are considered inefficient, because the execution times of the latter grow much more rapidly as the problem size increases.

A problem is called NP (nondeterministic polynomial) if its solution can be guessed and verified in polynomial time; nondeterministic means that no particular rule is followed to make the guess. If a problem is NP and all other NP problems are polynomial-time reducible to it, the problem is NP-complete. Thus, finding an efficient algorithm for any NP-complete problem implies that an efficient algorithm can be found for all such problems, since any problem belonging to this class can be recast into any other member of the class. It is not known whether any polynomial-time algorithms will ever be found for NP-complete problems, and determining whether these problems are tractable or intractable remains one of the most important questions in theoretical computer science. When an NP-complete problem must be solved, one approach is to use a polynomial algorithm to approximate the solution; the answer thus obtained will not necessarily be optimal but will be reasonably close.

In computational complexity theory, a problem is **NP-complete** when:

1. it is a problem for which the correctness of each solution can be verified quickly (namely, in polynomial time) and a brute-force search algorithm can find a solution by trying all possible solutions.
2. the problem can be used to simulate every other problem for which we can verify quickly that a solution is correct. In this sense, NP-complete problems are the hardest of the problems to which solutions can be verified quickly. If we could find solutions of some NP-complete problem quickly, we could quickly find the solutions of every other problem to which a given solution can be easily verified.

The name "NP-complete" is short for "nondeterministic polynomial-time complete". In this name, "nondeterministic" refers to nondeterministic Turing machines, a way of mathematically formalizing the idea of a brute-force search algorithm. Polynomial time refers to an amount of time that is considered "quick" for a deterministic algorithm to check a single solution, or for a nondeterministic Turing machine to perform the whole search. "Complete" refers to the property of being able to simulate everything in the same complexity class.

More precisely, each input to the problem should be associated with a set of solutions of polynomial length, whose validity can be tested quickly (in polynomial time),[2] such that the output for any input is "yes" if the solution set is non-empty and "no" if it is empty. The complexity class of problems of this form is called NP, an abbreviation for "nondeterministic polynomial

time". A problem is said to be NP-hard if everything in NP can be transformed in polynomial time into it even though it may not be in NP. Conversely, a problem is NP-complete if it is both in NP and NP-hard. The NP-complete problems represent the hardest problems in NP. If some NP-complete problem has a polynomial time algorithm, all problems in NP do. The set of NP-complete problems is often denoted by **NP-C** or **NPC**.

Although a solution to an NP-complete problem can be *verified* "quickly", there is no known way to *find* a solution quickly. That is, the time required to solve the problem using any currently known algorithm increases rapidly as the size of the problem grows. As a consequence, determining whether it is possible to solve these problems quickly, called the P versus NP problem, is one of the fundamental unsolved problems in computer science today.

While a method for computing the solutions to NP-complete problems quickly remains undiscovered, computer scientists and programmers still frequently encounter NP-complete problems. NP-complete problems are often addressed by using heuristic methods and approximation algorithms.

Boolean satisfiability problem

In logic and computer science, the **Boolean satisfiability problem** (sometimes called **propositional satisfiability problem** and abbreviated **SATISFIABILITY**, **SAT** or **B-SAT**) is the problem of determining if there exists an interpretation that satisfies a given Boolean formula. In other words, it asks whether the variables of a given Boolean formula can be consistently replaced by the values TRUE or FALSE in such a way that the formula evaluates to TRUE. If this is the case, the formula is called *satisfiable*. On the other hand, if no such assignment exists, the function expressed by the formula is FALSE for all possible variable assignments and the formula is *unsatisfiable*. For example, the formula "*a* AND NOT *b*" is satisfiable because one can find the values *a* = TRUE and *b* = FALSE, which make (*a* AND NOT *b*) = TRUE. In contrast, "*a* AND NOT *a*" is unsatisfiable.

SAT is the first problem that was proved to be NP-complete; see Cook–Levin theorem. This means that all problems in the complexity class NP, which includes a wide range of natural decision and optimization problems, are at most as difficult to solve as SAT. There is no known algorithm that efficiently solves each SAT problem, and it is generally believed that no such algorithm exists; yet this belief has not been proved mathematically, and resolving the question of whether SAT has a polynomial-time algorithm is equivalent to the P versus NP problem, which is a famous open problem in the theory of computing.

Nevertheless, as of 2007, heuristic SAT-algorithms are able to solve problem instances involving tens of thousands of variables and formulas consisting of millions of symbols,[1] which is sufficient for many practical SAT problems from, e.g., artificial intelligence, circuit design,[2] and automatic theorem proving.

What is Non-deterministic Polynomial Time?

NP, for **n**on-deterministic **p**olynomial time, is one of the best-known complexity classes in theoretical computer science. A decision problem (a problem that has a yes/no answer) is said to

be in NP if it is solvable in polynomial time by a *non-deterministic*Turing machine. Equivalently, and more intuitively, a decision problem is in NP if, if the answer is yes, a proof can be verified by a Turing machine in polynomial time.

A Practical Example

An easy example of this is Sudoku. To phrase it as a decision problem, you would say something like, "Given a sudoku puzzle, does it have a solution?" It may take a long time to answer that question (because you have to solve the puzzle), but if someone gives you a solution you can very quickly verify that the solution is correct. That's what NP is -- the set of all decision problems where you can efficiently verify the answer.

In contrast, P (polynomial time) is the set of all decision problems which can be solved in polynomial time by a Turing machine. Roughly speaking, if a problem is in P, then it's considered tractable, i.e. there exists an algorithm that can solve it in a reasonable amount of time on a computer. If a problem is not in P, then it's said to be intractable, meaning that for large values it would take far too long for even the best supercomputer to solve it - in some cases, this means millions or even billions of years.

In Computer Science and Machine Learning

An open question in theoretical computer science is whether P = NP. If P and NP were shown to be identical, then that would mean that all of the problems in NP are also in P -- i.e. that some algorithm exists to solve any given problem in NP quickly. This would be earth shattering because there are a lot of problems in NP that it would be convenient to be able to solve quickly, and also because all of cryptography is based on the assumption that certain problems in NP are too difficult for anyone to solve in a reasonable amount of time. Conversely, if we had a proof that P ≠ NP, then we would know for certain that there are certain problems (the NP-complete problems) which have no quick solutions.

Examples of NP problems include:

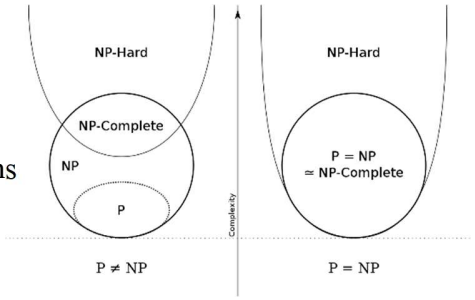
- Calculating the greatest common divisor of a number (this is in P)
- Linear programming (this is in P)
- The subgraph isomorphism problem: given a graph G and a graph H, does G contain a subgraph that's isomorphic to H? (this is in NP)
- Given a set of points, does there exist some route that goes through each point of a length less than k? (Travelling Salesman Problem; this is in NP-complete)

https://deepai.org/machine-learning-glossary-and-terms/nondeterministic-polynomial-time

NP-hardness

In computational complexity theory, **NP-hardness** (non-deterministic polynomial-time hardness) is the defining property of a class of problems that are informally "at least as hard as the hardest problems in NP". A simple example of an NP-hard problem is the subset sum problem.

A more precise specification is: a problem *H* is NP-hard when every problem *L* in NP can be reduced in polynomial time to *H*; that is, assuming a solution for *H* takes 1 unit time, *H*'s solution can be used to solve *L* in polynomial time.[1][2] As a consequence, finding a polynomial time algorithm to solve any NP-hard problem would give polynomial time algorithms for all the problems in NP. As it is suspected that P≠NP, it is unlikely that such an algorithm exists.[3]



It is suspected that there are no polynomial-time algorithms for NP-hard problems, but that has not been proven.[4] Moreover, the class P, in which all problems can be solved in polynomial time, is contained in the NP class.

https://en.wikipedia.org/wiki/NP-hardness#/media/File:P_np_np-complete_np-hard.svg

An example of an NP-hard problem is the decision subset sum problem: given a set of integers, does any non-empty subset of them add up to zero? That is a decision problem and happens to be NP-complete. Another example of an NP-hard problem is the optimization problem of finding the least-cost cyclic route through all nodes of a weighted graph. This is commonly known as the travelling salesman problem.[7]

There are decision problems that are *NP-hard* but not *NP-complete* such as the halting problem. That is the problem which asks "given a program and its input, will it run forever?" That is a *yes*/*no* question and so is a decision problem. It is easy to prove that the halting problem is NP-hard but not NP-complete. For example, the Boolean satisfiability problem can be reduced to the halting problem by transforming it to the description of a Turing machine that tries all truth value assignments and when it finds one that satisfies the formula it halts and otherwise it goes into an infinite loop. It is also easy to see that the halting problem is not in *NP* since all problems in NP are decidable in a finite number of operations, but the halting problem, in general, is undecidable. There are also NP-hard problems that are neither *NP-complete* nor *Undecidable*. For instance, the language of true quantified Boolean formulas is decidable in polynomial space, but not in non-deterministic polynomial time (unless NP = PSPACE).

NP-naming convention

NP-hard problems do not have to be elements of the complexity class NP. As NP plays a central role in computational complexity, it is used as the basis of several classes:

NP

Class of computational decision problems for which any given *yes*-solution can be verified as a solution in polynomial time by a deterministic Turing machine (or *solvable* by a *non-deterministic* Turing machine in polynomial time).

NP-hard

Class of problems which are at least as hard as the hardest problems in NP. Problems that are NP-hard do not have to be elements of NP; indeed, they may not even be decidable.

NP-complete

Class of decision problems which contains the hardest problems in NP. Each NP-complete problem has to be in NP.

NP-easy

At most as hard as NP, but not necessarily in NP.

NP-equivalent

Decision problems that are both NP-hard and NP-easy, but not necessarily in NP.

NP-intermediate

If P and NP are different, then there exist decision problems in the region of NP that fall between P and the NP-complete problems. (If P and NP are the same class, then NP-intermediate problems do not exist because in this case every NP-complete problem would fall in P, and by definition, every problem in NP can be reduced to an NP-complete problem.)

An Optimal Task Assignment Strategy in Cloud-Fog Computing Environment

With the advent of the Internet of Things era, more and more emerging applications need to provide real-time interactive services. Although cloud computing has many advantages, the massive expansion of the Internet of Things devices and the explosive growth of data may induce network congestion and add network latency. Cloud-fog computing processes some data locally on edge devices to reduce the network delay. This paper investigates the optimal task assignment strategy by considering the execution time and operating costs in a cloud-fog computing environment. Linear transformation techniques are used to solve the nonlinear mathematical programming model of the task assignment problem in cloud-fog computing systems. The proposed method can determine the globally optimal solution for the task assignment problem based on the requirements of the tasks, the processing speed of nodes, and the resource usage cost of nodes in cloud-fog computing systems.

Introduction

In the era of the Internet of Things (IoT), many emerging applications need to provide real-time responses and interactions. According to the report by market insights firm IoT Analytics, global IoT device connections are estimated to surpass non-IoT device connections and reach 11.7 billion in 2020 [10]. The International Data Corporation (IDC) predicts that 75% of 55.7 billion devices worldwide will be connected to an IoT platform by 2025. IDC also estimates that the data generated from the IoT devices will grow from 18.3 ZB in 2019 to 73.1 ZB by 2025 [11].

In various industries, such as transportation, oil and gas, manufacturing, mining, and utilities, a short response time plays a vital role in improving the output, boosting service levels, and increasing the safety. The data sensed from the IoT devices often requires a rapid and real-time analysis of the data. Consequently, an appropriate infrastructure must be designed to deal with the rapid growth of the IoT data for making a timely and correct decision during event detection [12].

Cisco [13] listed the main requirements for the processing of the IoT data; a well-designed task assignment strategy can satisfy the following requirements:

- latency decrement
- network bandwidth conservation
- data movement to the best place for processing

Although cloud computing has many advantages, the massive expansion of the IoT devices and the explosive growth of data may induce network congestion and add network latency. Conventional cloud computing architectures that transmit all data from the network edge to the central data center for processing cannot meet all of the above requirements. Transmitting the huge amount of data from the IoT devices to the cloud imposes a significantly heavy burden on network performance. This situation also results in unreliable network latency or uncertain response time for end-users [13,14].

Edge computing has a decentralized architecture that assigns processing tasks to the edge in the network to reduce the network delay. Fog computing transforms network edge devices into parts of a distributed computing architecture to implement IoT applications such as medical and healthcare, building and home automation, traffic control, environmental monitoring, energy management, transportation networks, etc. [12,13]. Compared to pure cloud computing, edge computing and fog computing perform better within the aspects of data transmission speed, privacy and security, limited bandwidths, and data control [15]. Due to the advancements in information technology, conventional network edge devices—for instance, routers, gateways, workstations, and personal computers—have become increasingly powerful in the context of the processing capability, storage space, and communication capability. The resources not only can be utilized by their owners but help to push data handling to the network edge [14].

This paper focuses on the optimal task assignment strategy that minimizes the execution time and operating costs in a cloud-fog computing environment. Nguyen et al. [13] constructed a nonlinear mathematical programming model to treat the task assignment problem in a cloud-fog computing environment for the IoT. The model consists of an objective function that involves a parameter to control the trade-off between task completion time and total cost. They also developed evolutionary algorithms to solve the problem. However, their methods cannot guarantee the global optimality of the obtained solution. This study proposes a method to transform the nonlinear problem into a linear model and then guarantees to find a globally optimal solution of the task assignment problem.

https://www.mdpi.com/2076-3417/11/4/1909/htm#sec2-applsci-11-01909

Travelling salesman problem

The **travelling salesman problem** (also called the **travelling salesperson problem** or **TSP**) asks the following question: "Given a list of cities and the distances between each pair of cities, what is the shortest possible route that visits each city exactly once and returns to the origin city?" It is

an NP-hard problem in combinatorial optimization, important in theoretical computer science and operations research.

The travelling purchaser problem and the vehicle routing problem are both generalizations of TSP.

In the theory of computational complexity, the decision version of the TSP (where given a length L, the task is to decide whether the graph has a tour of at most L) belongs to the class of NP-complete problems. Thus, it is possible that the worst-case running time for any algorithm for the TSP increases superpolynomially (but no more than exponentially) with the number of cities.

The problem was first formulated in 1930 and is one of the most intensively studied problems in optimization. It is used as a benchmark for many optimization methods. Even though the problem is computationally difficult, many heuristics and exact algorithms are known, so that some instances with tens of thousands of cities can be solved completely and even problems with millions of cities can be approximated within a small fraction of 1%.[1]

The TSP has several applications even in its purest formulation, such as planning, logistics, and the manufacture of microchips. Slightly modified, it appears as a sub-problem in many areas, such as DNA sequencing. In these applications, the concept *city* represents, for example, customers, soldering points, or DNA fragments, and the concept *distance* represents travelling times or cost, or a similarity measure between DNA fragments. The TSP also appears in astronomy, as astronomers observing many sources will want to minimize the time spent moving the telescope between the sources; in such problems, the TSP can be embedded inside an optimal control problem. In many applications, additional constraints such as limited resources or time windows may be imposed.

Minimum halls required for class scheduling

Given **N** lecture timings, with their start time and end time (both inclusive), the task is to find the minimum number of halls required to hold all the classes such that a single hall can be used for only one lecture at a given time. Note that the maximum end time can be 10^5.
Examples:

Input: lectures[][] = {{0, 5}, {1, 2}, {1, 10}}
Output: 3
All lectures must be held in different halls because
at time instance 1 all lectures are ongoing.
Input: lectures[][] = {{0, 5}, {1, 2}, {6, 10}}
Output: 2
Approach:

- Assuming that time T starts with 0. The task is to find the maximum number of lectures that are ongoing at a particular instance of time. This will give the minimum number of halls required to schedule all the lectures.

- To find the number of lectures ongoing at any instance of time. Maintain a prefix_sum[] which will store the number of lectures ongoing at any instance of time t. For any lecture with timings between [s, t], do prefix_sum[s]++ and prefix_sum[t + 1]–.
- Afterward, the cumulative sum of this prefix array will give the count of lectures going on at any instance of time.
- The maximum value for any time instant t in the array is the minimum number of halls required.

Below is the implementation of the above approach:

```
// C# implementation of the approach
using System;

class GFG
{
static int MAX = 100001;

// Function to return the minimum
// number of halls required
static int minHalls(int [,]lectures, int n)
{

    // Array to store the number of
    // lectures ongoing at time t
    int []prefix_sum = new int[MAX];

    // For every lecture increment start
    // point s decrement (end point + 1)
    for (int i = 0; i < n; i++)
    {
        prefix_sum[lectures[i,0]]++;
        prefix_sum[lectures[i,1] + 1]--;
    }

    int ans = prefix_sum[0];

    // Perform prefix sum and update
    // the ans to maximum
    for (int i = 1; i < MAX; i++)
    {
        prefix_sum[i] += prefix_sum[i - 1];
        ans = Math.Max(ans, prefix_sum[i]);
    }
    return ans;
}

// Driver code
public static void Main(String[] args)
{
    int [,]lectures = {{ 0, 5 },
                       { 1, 2 },
                       { 1, 10 }};
    int n = lectures.GetLength(0);

    Console.WriteLine(minHalls(lectures, n));
}
}
 // This code is contributed by 29AjayKumar
```

Output

3

Time Complexity: O(MAX)

Auxiliary Space: O(MAX)

where MAX is 100001.

Another Approach:

The above approach works when MAX is limited to 10^5. When the limits of MAX are extended up to 10^9, we cannot use above approach due to Memory Limit and Time Limit Constraints.

So, we think in a different dimension, towards sorting and cumulative sum. Store the lecture time (start time and end time) in chronological order, +1 denoting the start time of a lecture and -1 denoting the end time of a lecture. Then apply the concept of cumulative sum, this gives the maximum number of lectures being conducted at a time. This gives the bare minimum number of halls that are required.

Algorithm:

- Initialize a vector of pair, *Time*, the first value of which indicates the entry or exit time of lecture and the second value denotes whether the lecture starts or ends.
- Traverse the lectures vector and store the values in the vector *Time*.
- Sort the vector *Time*.
- Maintain two variables *answer* = 1, and *sum* = 0. *answer* denotes the final answer to be returned and *sum* denotes the number of lectures going on at a particular time. Note that *answer* is initialized as 1 because at least 1 hall is required for a lecture to be conducted.
- Traverse the *Time* vector, add the second value of the pair into *sum* and update the *answer* variable.
- Return *answer*.

Below is the implementation of the above approach:

```
// C# implementation of the above approach
using System;
using System.Collections.Generic;

public class Pair<T, U> {
  public Pair() {}

  public Pair(T first, U second)
  {
    this.First = first;
    this.Second = second;
  }

  public T First
  {
    get;
    set;
  }
  public U Second
  {
    get;
    set;
  }
};
class GFG {
```

```
  public static int cmp(Pair<int, int> a,
                        Pair<int, int> b)
  {
    if(a.First < b.First)
      return -1;
    else
      return 0;
  }
  // Function to return the minimum
  // number of halls required
  static int minHalls(int[, ] lectures, int n)
  {

    // Initialize a vector of pair, Time, first value
    // indicates the time of entry or exit of a lecture
    // second value denotes whether the lecture starts
    // or ends
    List<Pair<int, int> > Time
      = new List<Pair<int, int> >();
    // Store the lecture times
    for (int i = 0; i < n; i++) {
      Time.Add(new Pair<int, int>(lectures[i, 0], 1));
      Time.Add(new Pair<int, int>(lectures[i, 1], -1));
    }

    // Sort the vector
    Time.Sort(cmp);

    int answer = 0, sum = 0;

    // Traverse the Time vector and Update sum and
    // answer variables
    for (int i = 0; i < 2*n; i++) {
      sum += Time[i].Second;
      answer = Math.Max(answer, sum);
    }

    // Return the answer
    return answer;
  }

  // Driver Code
  static public void Main(String[] args)
  {
    int[,] lectures = { { 0, 5 }, { 1, 2 }, { 1, 10 } };
    int n = lectures.GetLength(0);

    Console.WriteLine(minHalls(lectures, n));
  }
}

// This code is contributed by Abhijeet Kumar(abhijeet19403)
```

Output

3

Time Complexity: O(n*log(n))

Auxiliary Space: O(n)

Note that instead of vector of pair, one can use map or priority queue, the time complexity and space complexity would remain the same.

https://www.geeksforgeeks.org/minimum-halls-required-for-class-scheduling/

Definition

Design Space Optimization (DSO) is a novel approach to searching large design spaces enabled by recent advancements in machine-learning. A marked departure from traditional Design Space Exploration (DSE), where engineers had to manually sweep very large problem spaces for design solutions, DSO is a generative optimization paradigm that uses *reinforcement-learning* (RL) technology to *autonomously* search design spaces for optimal solutions.

Chip Design: A Vast Search Space

Today, AI can interact with humans through natural language, identify bank fraud and protect computer networks, drive cars around city streets, and play intelligent games like chess and Go. Chip design is a very large space of potential solutions, trillions of times larger than, for example, the game of Go.

A chip design workflow typically consumes and generates terabytes of highly dimensional data compartmentalized and fragmented across many separately optimized silos. This data can include:

- **Design Inputs.** RTL models, power activity, netlists, floorplans, hierarchy choices
- **Technology Inputs.** Device and circuit characteristics, parasitic models, DFM models, metal stacks, design rules
- **Library Inputs.** Cell architecture, cell schematics, timing/power attributes, physical characteristics
- **Tool Inputs.** Design flow, test vectors, hierarchical partitioning, heuristic settings

---

Traditional Design Space Exploration: A Manual Process

Searching this vast space is a very labor-intensive effort, typically requiring many weeks of experimentation, and often guided by past experiences and tribal knowledge. To create an optimal design recipe, engineers must ingest volumes of high-velocity data and make complex decisions on the fly with incomplete analysis, often leading to decision fatigue and over-constraining of their design. This manual, iterative, *Design Space Exploration (DSE)* process has partitioned the problem to manage its complexity. By consequence, the exploration of choices in design workflows has been severely limited, and designers' productivity has been impacted by a large volume of small decisions.

In today's hypercompetitive markets and stringent silicon manufacturing requirements, the difference between a good recipe and an optimal recipe can be 100s of MHz of performance; hours of battery life; or millions of dollars in design costs.

Machine Learning — Everywhere!

How to enable self-optimizing design platforms for better end-to-end results.

Download White Paper

DSO: Autonomously Searching & Optimizing Design Spaces

DSO massively scales the exploration of choices in chip design workflows, while automating a high volume of less consequential decisions.

The DSO Agent analyzes large data streams generated by design tools and uses them to make optimization decisions, observing how a design evolves over time and adjusting design choices, technology parameters, and workflows to guide the search process towards multi-dimensional optimization objectives. Searches can be executed at massive scale, with RL-driven decision engines autonomously operating tens-to-thousands of exploration vectors and ingesting gigabytes of high-velocity design analysis data – all in real-time.

A key characteristic of AI-driven process optimization, DSO automates less consequential decisions, like tuning simulation engine settings, or orchestrating experiments, relieving designers of menial tasks and allowing teams to operate at a near-expert level. Once a DSO system builds up models on a given design workflow, the learnings can be shared and applied with high effectiveness across projects and design teams.

AI-grade DSO automation opens up a new growth trajectory for the semiconductor industry, enable companies to build new products and harvest new markets, and design teams to grow and deliver the innovations of tomorrow.

B) Search the Internet for*: Benchmark functions to evaluate the Optimization Methods proposed and give a brief description and their basic difficulty & property of their testing method.*

What is benchmark functions for optimization?

In literature, benchmark test functions have been used for **evaluating performance of metaheuristic algorithms**. Algorithms that perform well on a set of numerical optimization problems are considered as effective methods for solving real-world problems.

What is the function of benchmark test?

It can used **to identify a student's strengths and weaknesses**. Benchmark Testing typically occurs in reading and math. Results indicate where a student is according to a national percentile in the areas of: Reading.

What is optimization function in machine learning?

Optimization is **the process where we train the model iteratively that results in a maximum and minimum function evaluation**. It is one of the most important phenomena in Machine Learning to get better results.

What is a benchmark explain with examples?

Benchmarking is **a process of measuring the performance of a company's products, services, or processes against those of another business considered to be the best in the industry, aka "best in class."** The point of benchmarking is to identify internal opportunities for improvement.

What is a benchmark and why is it important?

Benchmarking is **the practice of comparing a company's metrics to other businesses to analyze what's effective in their respective industries**. Implementing this process can lead to various benefits, including improved efficiency and increased sales.

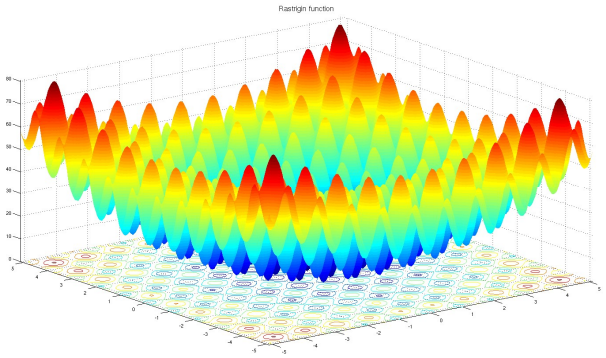
What are three purposes of benchmark test?

The National Benchmark Tests (NBTs) are assessments for first-year applicants into higher education institutions. The NBTs were designed **to measure a writer's ability to transfer understanding of Academic Literacy, Quantitative Literacy and Mathematics to the demands of tertiary coursework**.

What are the four types of benchmark?

There are four main types of benchmarking: **internal, external, performance, and practice**.

Rastrigin function



In mathematical optimization, the **Rastrigin function** is a non-convex function used as a performance test problem for optimization algorithms. It is a typical example of non-linear multimodal function. It was first proposed in 1974 by Rastrigin[1] as a 2-dimensional function and has been generalized by Rudolph.[2] The generalized version was popularized by Hoffmeister & Bäck[3] and Mühlenbein et al.[4] Finding the minimum of this function is a fairly difficult problem due to its large search space and its large number of local minima.

9. FINDING/PRESENTING SEARCH ALGORITHMS BY SIMIATING NATURE

***A) Search the Internet**: Search algorithms that mimic nature. Give a brief description of their function and list the sources/links.*

***B) Search the Internet**: Their applications in your specialty. Give a brief description of the application and list the sources/links.*

Ant Colony (AC)

What is ant colony algorithm?

The ant colony algorithm is **an algorithm for finding optimal paths that is based on the behavior of ants searching for food**. At first, the ants wander randomly. When an ant finds a source of food, it walks back to the colony leaving "markers" (pheromones) that show the path has food.

https://mathworld.wolfram.com/AntColonyAlgorithm.html

Where is ant colony algorithm used?

Ant colony optimization is a probabilistic technique for finding optimal paths. In computer science and researches, the ant colony optimization algorithm is used for **solving different computational problems**.

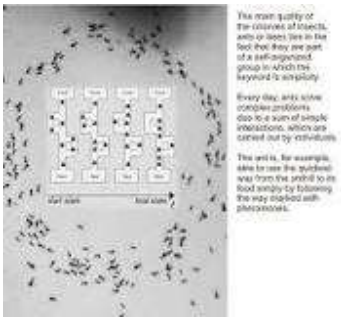
https://towardsdatascience.com/the-inspiration-of-an-ant-colony-optimization-f377568ea03f

https://www.ijcaonline.org/volume5/number4/pxc3871286.pdf

What is ant colony optimization used for?

Ant colony optimization (ACO) is a population-based metaheuristic that can be used **to find approximate solutions to difficult optimization problems**. In ACO, a set of software agents called artificial ants search for good solutions to a given optimization problem.

http://www.scholarpedia.org/article/Ant_colony_optimization



How does ACO algorithm work?

In the first step of each iteration, each ant stochastically constructs a solution, i.e. the order in which the edges in the graph should be followed. In the second step, the paths found by the different ants are compared. The last step consists of updating the pheromone levels on each edge.

Artificial Bee Colony (ABC)

How does artificial bee colony algorithm work?

In the ABC algorithm, **the bees in a colony are divided into three groups: employed bees (forager bees), onlooker bees (observer bees), and scouts**. For each food source, there is only one employed bee. That is to say, the number of employed bees is equal to the number of food sources.

https://www.sciencedirect.com/topics/computer-science/artificial-bee-colony-algorithm

What do you optimize in artificial bee colony algorithm?

The Artificial Bee Colony (ABC) algorithm is a swarm based meta-heuristic algorithm that was introduced by Karaboga in 2005 (Karaboga, 2005) for optimizing numerical problems. It was inspired by the intelligent foraging behavior of honey bees.

http://www.scholarpedia.org/article/Artificial_bee_colony_algorithm

Can you ai a bee?

Artificial insemination (AI) of queen bees is a growing practice worldwide. It's an important practice for maintaining the health and quality of your bee stock.

https://microscopes.com.au/blogs/news/artificial-insemination-of-queen-bees-what-to-look-for-in-a-microscope

Frog Leaping

The shuffled frog leaping algorithm is **a population-based approach for a heuristic search in optimization problems**. The algorithm consists of a set of virtual frogs partitioned into several groups called "memeplexes". However, After some optimization runs frogs position's become closer in each memeplex.

https://ieeexplore.ieee.org/abstract/document/6936975

https://www.tandfonline.com/doi/abs/10.1080/03052150500384759

https://towardsdatascience.com/a-survey-on-shuffled-frog-leaping-algorithm-d309d0cf7503

Cuckoo search

In operations research, **cuckoo search** is an optimization algorithm developed by Xin-She Yang and Suash Deb in 2009.[16][17] It was inspired by the obligate brood parasitism of some cuckoo species by laying their eggs in the nests of host birds of other species. Some host birds can engage direct conflict with the intruding cuckoos. For example, if a host bird discovers the eggs are not their own, it will either throw these alien eggs away or simply abandon its nest and build a new nest elsewhere. Some cuckoo species such as the New World brood-parasitic Tapera have evolved in such a way that female parasitic cuckoos are often very specialized in the mimicry in colors and pattern of the eggs of a few chosen host species.[18] Cuckoo search idealized such breeding behavior, and thus can be applied for various optimization problems. It has been shown that cuckoo search is a special case of the well-known $(\mu + \lambda)$-evolution strategy.

What are 3 idealized rules for cuckoo search algorithm?

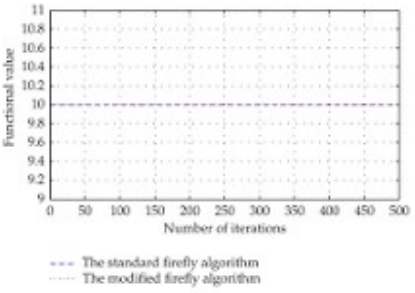
For simplicity in describing our new Cuckoo Search, we now use the following three idealized rules: 1) Each cuckoo lays one egg at a time, and dump its egg in randomly chosen nest; 2) The best nests with high quality of eggs will carry over to the next generations; 3) The number of available host nests is fixed, and ...

https://www.cs.tufts.edu/comp/150GA/homeworks/hw3/_reading7%20Cuckoo%20search.pdf

https://www.youtube.com/watch?v=46we_zNBhKA

Firefly Selection

How does the firefly algorithm work?



Firefly algorithm is one of the new metaheuristic algorithms for optimization problems. The algorithm is inspired by the flashing behavior of fireflies. In the algorithm, **randomly generated solutions will be considered as fireflies, and brightness is assigned depending on their performance on the objective function**.

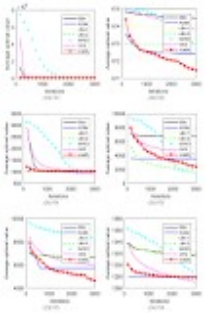
https://www.hindawi.com/journals/jam/2012/467631/

What are some limitations of the Firefly model?

Firefly algorithm (FA) is one of the swarm intelligence algorithms, which is proposed by Yang in 2008. The standard FA has some disadvantages, such as high computational time complexity, slow convergence speed and so on.

https://ieeexplore.ieee.org/document/8830381

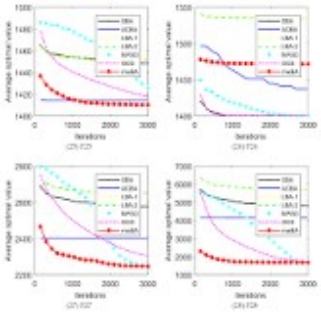
Bat Search



What is bat optimization?

A bat algorithm (BA) is **a heuristic algorithm that operates by imitating the echolocation behavior of bats to perform global optimization**. The BA is widely used in various optimization problems because of its excellent performance.

https://www.mdpi.com/2227-7390/7/2/135



How does bat algorithm work?

A bat algorithm (BA) is a heuristic algorithm that operates by **imitating the echolocation behavior of bats to perform global optimization**. The BA is widely used in various optimization problems because of its excellent performance.

What are the applications of BAT algorithm?

The upper layer objective function is solved using bat algorithm and the obtained solution is further used to optimize the lower layer. Flowchart for the traffic signal optimization problem using bat algorithm. The frequency of bats where n is the number of nodes in the network.

https://www.hindawi.com/journals/acisc/2019/9864090/

Why is the bats program important?

By eating pests, bats save producers more than $1 billion per year in crop damage and pesticide costs. More than 30 bat species are listed under the Endangered Species Act and

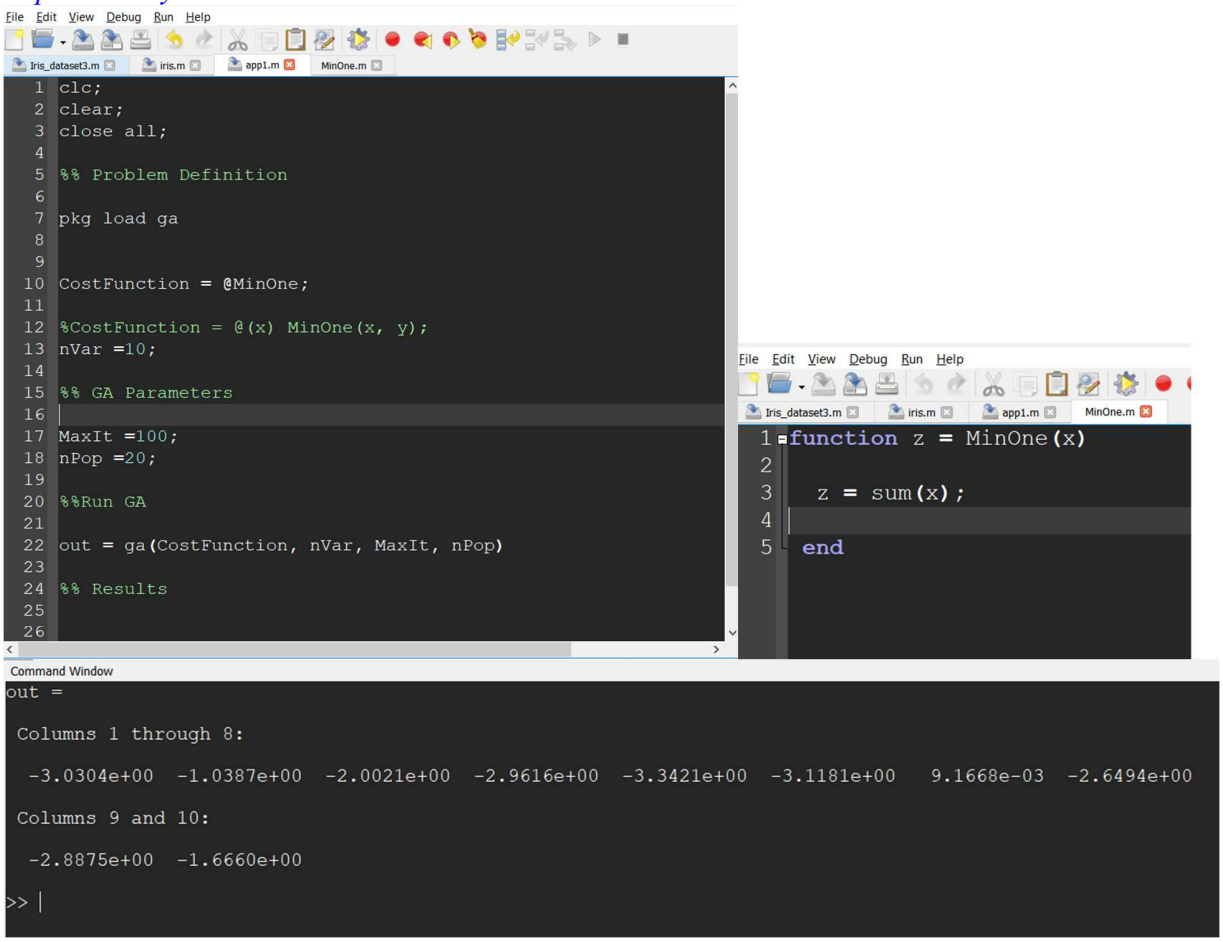
threatened by White-Nose Syndrome , habitat loss, pesticide use, climate change, damage to roost and forage sites and many other fears.

https://www.usda.gov/media/blog/2021/10/27/celebrating-special-powers-bats

10. FINDING/UNDERSTANDING THE ONLINE INSTRUCTIONS FOR USING BASIC GA GENETICAL ALGORITHM

***A) Search the Internet**: The instructions and Watch the videos with other examples of functions that you will implement:*

1) Binary – Min One, f(x) = SUM(xi), i=1:n => x=0*
https://www.youtube.com/watch?v=ICzcG0ORv6I



2) Sphere benchmark function, f(x) = SUM(xi^2), i=1:n => x=0*
https://www.youtube.com/watch?v=FuWPDQz6Oe0

Matlab

- Introductory article on genetic algorithms
  https://www.mathworks.com/help/gads/what-is-the-genetic-algorithm.html

- Terminology used in genetic algorithm technology https://www.mathworks.com/help/gads/some-genetic-algorithm-terminology.html
- Detailed outline of the stages of the algorithm and how exactly they are implemented in Matlab https://www.mathworks.com/help/gads/how-the-genetic-algorithm-works.html
- GA command invocation and arguments https://www.mathworks.com/help/gads/ga.html
- Options of the GA command https://www.mathworks.com/help/gads/genetic-algorithm-options.html

YouTube

- Binary Genetic Algorithm in Matlab – Part A (Practical Genetic Algorithm Series) https://www.youtube.com/watch?v=ICzcG0ORv6I
- Binary Genetic Algorithm in Matlab – Part b (Practical Genetic Algorithm Series) https://www.youtube.com/watch?v=pW39nKyYlN4
- Real-Coded Genetic Algorithm in MATLAB (Practical Genetic Algorithms Series) https://www.youtube.com/watch?v=FuWPDQz6Oe0
- Genetic Algorithm: General Concept, Matlab Code, and Example https://www.youtube.com/watch?v=MLPHLm56inQ

11. SEARCH FOR VARIATIONS OF THE BASIC GENETIC ALGORITHM

A) Search the Internet*: Variations (modifications) of GA and describe for 1-2 of these variations how they differ from the basic algorithm and what is the benefit of these changes:*

Variants

Chromosome representation

The simplest algorithm represents each chromosome as a bit string. Typically, numeric parameters can be represented by integers, though it is possible to use floating point representations. The floating point representation is natural to evolution strategies and evolutionary programming. The notion of real-valued genetic algorithms has been offered but is really a misnomer because it does not really represent the building block theory that was proposed by John Henry Holland in the 1970s. This theory is not without support though, based on theoretical and experimental results (see below). The basic algorithm performs crossover and mutation at the bit level. Other variants treat the chromosome as a list of numbers which are indexes into an instruction table, nodes in a linked list, hashes, objects, or any other imaginable data structure. Crossover and mutation are performed so as to respect data element boundaries. For most data types, specific variation operators can be designed. Different chromosomal data types seem to work better or worse for different specific problem domains.

When bit-string representations of integers are used, Gray coding is often employed. In this way, small changes in the integer can be readily affected through mutations or crossovers. This has been found to help prevent premature convergence at so-called *Hamming walls*, in which too many simultaneous mutations (or crossover events) must occur in order to change the chromosome to a better solution.

Other approaches involve using arrays of real-valued numbers instead of bit strings to represent chromosomes. Results from the theory of schemata suggest that in general the smaller the alphabet, the better the performance, but it was initially surprising to researchers that good results were obtained from using real-valued chromosomes. This was explained as the set of real values in a finite population of chromosomes as forming a *virtual alphabet* (when selection and recombination are dominant) with a much lower cardinality than would be expected from a floating point representation.[19][20]

An expansion of the Genetic Algorithm accessible problem domain can be obtained through more complex encoding of the solution pools by concatenating several types of heterogenously encoded genes into one chromosome.[21] This particular approach allows for solving optimization problems that require vastly disparate definition domains for the problem parameters. For instance, in problems of cascaded controller tuning, the internal loop controller structure can belong to a conventional regulator of three parameters, whereas the external loop could implement a linguistic controller (such as a fuzzy system) which has an inherently different description. This particular form of encoding requires a specialized crossover mechanism that recombines the chromosome by section, and it is a useful tool for the modelling and simulation of complex adaptive systems, especially evolution processes.

Elitism

A practical variant of the general process of constructing a new population is to allow the best organism(s) from the current generation to carry over to the next, unaltered. This strategy is known as *elitist selection* and guarantees that the solution quality obtained by the GA will not decrease from one generation to the next.[22]

Parallel implementations

Parallel implementations of genetic algorithms come in two flavors. Coarse-grained parallel genetic algorithms assume a population on each of the computer nodes and migration of individuals among the nodes. Fine-grained parallel genetic algorithms assume an individual on each processor node which acts with neighboring individuals for selection and reproduction. Other variants, like genetic algorithms for online optimization problems, introduce time-dependence or noise in the fitness function.

Adaptive GAs

Genetic algorithms with adaptive parameters (adaptive genetic algorithms, AGAs) is another significant and promising variant of genetic algorithms. The probabilities of crossover (pc) and mutation (pm) greatly determine the degree of solution accuracy and the convergence speed that genetic algorithms can obtain. Instead of using fixed values of *pc* and *pm*, AGAs utilize the population information in each generation and adaptively adjust the *pc* and *pm* in order to maintain the population diversity as well as to sustain the convergence capacity. In AGA (adaptive genetic algorithm),[23] the adjustment of *pc* and *pm* depends on the fitness values of the solutions. In *CAGA* (clustering-based adaptive genetic algorithm),[24] through the use of clustering analysis to judge the optimization states of the population, the adjustment of *pc* and *pm* depends on these

optimization states. It can be quite effective to combine GA with other optimization methods. A GA tends to be quite good at finding generally good global solutions, but quite inefficient at finding the last few mutations to find the absolute optimum. Other techniques (such as simple hill climbing) are quite efficient at finding absolute optimum in a limited region. Alternating GA and hill climbing can improve the efficiency of GA[*citation needed*] while overcoming the lack of robustness of hill climbing.

This means that the rules of genetic variation may have a different meaning in the natural case. For instance – provided that steps are stored in consecutive order – crossing over may sum a number of steps from maternal DNA adding a number of steps from paternal DNA and so on. This is like adding vectors that more probably may follow a ridge in the phenotypic landscape. Thus, the efficiency of the process may be increased by many orders of magnitude. Moreover, the inversion operator has the opportunity to place steps in consecutive order or any other suitable order in favour of survival or efficiency.[25]

A variation, where the population as a whole is evolved rather than its individual members, is known as gene pool recombination.

A number of variations have been developed to attempt to improve performance of GAs on problems with a high degree of fitness epistasis, i.e. where the fitness of a solution consists of interacting subsets of its variables. Such algorithms aim to learn (before exploiting) these beneficial phenotypic interactions. As such, they are aligned with the Building Block Hypothesis in adaptively reducing disruptive recombination. Prominent examples of this approach include the mGA,[26] GEMGA[27] and LLGA.

What are the various types of mutation techniques in genetic algorithm?

This mutation procedure, based on the biological point mutation, is called single point mutation. Other types are **inversion and floating point mutation**. When the gene encoding is restrictive as in permutation problems, mutations are swaps, inversions, and scrambles.

What are the basic variations of the evolutionary algorithm?

The main classes of EA in contemporary usage are (in order of popularity) **genetic algorithms (GAs), evolution strategies (ESs), differential evolution (DE) and estimation of distribution algorithms (EDAs)**.

What are the 3 different operations for genetic computing?

A genetic operator is an operator used in genetic algorithms to guide the algorithm towards a solution to a given problem. There are three main types of operators (**mutation, crossover and selection**), which must work in conjunction with one another in order for the algorithm to be successful.

Genetic operator

A **genetic operator** is an operator used in genetic algorithms to guide the algorithm towards a solution to a given problem. There are three main types of operators (mutation, crossover and

selection), which must work in conjunction with one another in order for the algorithm to be successful. Genetic operators are used to create and maintain genetic diversity (mutation operator), combine existing solutions (also known as chromosomes) into new solutions (crossover) and select between solutions (selection).[1] In his book discussing the use of genetic programming for the optimization of complex problems, computer scientist John Koza has also identified an 'inversion' or 'permutation' operator; however, the effectiveness of this operator has never been conclusively demonstrated and this operator is rarely discussed.[2][3]

Mutation (or mutation-like) operators are said to be *unary* operators, as they only operate on one chromosome at a time. In contrast, crossover operators are said to be *binary* operators, as they operate on two chromosomes at a time, combining two existing chromosomes into one new chromosome.[4]

Operators

Genetic variation is a necessity for the process of evolution. Genetic operators used in genetic algorithms are analogous to those in the natural world: survival of the fittest, or selection; reproduction (crossover, also called recombination); and mutation.

Selection

Selection operators give preference to better solutions (chromosomes), allowing them to pass on their 'genes' to the next generation of the algorithm. The best solutions are determined using some form of objective function (also known as a 'fitness function' in genetic algorithms), before being passed to the crossover operator. Different methods for choosing the best solutions exist, for example, fitness proportionate selection and tournament selection; different methods may choose different solutions as being 'best'. The selection operator may also simply pass the best solutions from the current generation directly to the next generation without being mutated; this is known as *elitism* or *elitist selection*.[1][5]

Crossover

Crossover is the process of taking more than one parent solutions (chromosomes) and producing a child solution from them. By recombining portions of good solutions, the genetic algorithm is more likely to create a better solution.[1] As with selection, there are a number of different methods for combining the parent solutions, including the *edge recombination operator* (ERO) and the 'cut and splice crossover' and 'uniform crossover' methods. The crossover method is often chosen to closely match the chromosome's representation of the solution; this may become particularly important when variables are grouped together as building blocks, which might be disrupted by a non-respectful crossover operator. Similarly, crossover methods may be particularly suited to certain problems; the ERO is generally considered a good option for solving the travelling salesman problem.[6]

Mutation

The mutation operator encourages genetic diversity amongst solutions and attempts to prevent the genetic algorithm converging to a local minimum by stopping the solutions becoming too close to

one another. In mutating the current pool of solutions, a given solution may change entirely from the previous solution. By mutating the solutions, a genetic algorithm can reach an improved solution solely through the mutation operator.[1] Again, different methods of mutation may be used; these range from a simple *bit mutation* (flipping random bits in a binary string chromosome with some low probability) to more complex mutation methods, which may replace genes in the solution with random values chosen from the uniform distribution or the Gaussian distribution. As with the crossover operator, the mutation method is usually chosen to match the representation of the solution within the chromosome.

Combining operators

While each operator acts to improve the solutions produced by the genetic algorithm working individually, the operators must work in conjunction with each other for the algorithm to be successful in finding a good solution. Using the selection operator on its own will tend to fill the solution population with copies of the best solution from the population. If the selection and crossover operators are used without the mutation operator, the algorithm will tend to converge to a local minimum, that is, a good but sub-optimal solution to the problem. Using the mutation operator on its own leads to a random walk through the search space. Only by using all three operators together can the genetic algorithm become a noise-tolerant hill-climbing algorithm, yielding good solutions to the problem.

12. GENETIC ALGORITHM (GA) APPLICATIONS IN SPECIALTY

A) Search the Internet*: Search the Internet for applications of GAs in your specialty. Give a brief description of the problem and solution and the reasons why GA is preferred:*

Feature selection

In machine learning and statistics, **feature selection**, also known as **variable selection**, **attribute selection** or **variable subset selection**, is the process of selecting a subset of relevant features (variables, predictors) for use in model construction. Feature selection techniques are used for several reasons:

- simplification of models to make them easier to interpret by researchers/users,[28]
- shorter training times,[29]
- to avoid the curse of dimensionality,[30]
- improve data's compatibility with a learning model class,[31]
- encode inherent symmetries present in the input space.[32][33][34][35]

The central premise when using a feature selection technique is that the data contains some features that are either *redundant* or *irrelevant*, and can thus be removed without incurring much loss of information.[9] *Redundant* and *irrelevant* are two distinct notions, since one relevant feature may be redundant in the presence of another relevant feature with which it is strongly correlated.[36]

Feature selection techniques should be distinguished from feature extraction.[11] Feature extraction creates new features from functions of the original features, whereas feature selection returns a subset of the features. Feature selection techniques are often used in domains where there are many features and comparatively few samples (or data points). Archetypal cases for the application of

feature selection include the analysis of written texts and DNA microarray data, where there are many thousands of features, and a few tens to hundreds of samples.

Computational creativity

Computational creativity (also known as **artificial creativity**, **mechanical creativity**, **creative computing** or **creative computation**) is a multidisciplinary endeavour that is located at the intersection of the fields of artificial intelligence, cognitive psychology, philosophy, and the arts (e.g., **computational art** as part of **computational culture**).

The goal of computational creativity is to model, simulate or replicate creativity using a computer, to achieve one of several ends:[37]

- To construct a program or computer capable of human-level creativity.
- To better understand human creativity and to formulate an algorithmic perspective on creative behavior in humans.
- To design programs that can enhance human creativity without necessarily being creative themselves.

The field of computational creativity concerns itself with theoretical and practical issues in the study of creativity. Theoretical work on the nature and proper definition of creativity is performed in parallel with practical work on the implementation of systems that exhibit creativity, with one strand of work informing the other.

The applied form of computational creativity is known as media synthesis.

Cryptanalysis

Cryptanalysis (from the Greek *kryptós*, "hidden", and *analýein*, "to analyze") refers to the process of analyzing information systems in order to understand hidden aspects of the systems.[38] Cryptanalysis is used to breach cryptographic security systems and gain access to the contents of encrypted messages, even if the cryptographic key is unknown.

In addition to mathematical analysis of cryptographic algorithms, cryptanalysis includes the study of side-channel attacks that do not target weaknesses in the cryptographic algorithms themselves, but instead exploit weaknesses in their implementation.

Even though the goal has been the same, the methods and techniques of cryptanalysis have changed drastically through the history of cryptography, adapting to increasing cryptographic complexity, ranging from the pen-and-paper methods of the past, through machines like the British Bombes and Colossus computers at Bletchley Park in World War II, to the mathematically advanced computerized schemes of the present. Methods for breaking modern cryptosystems often involve solving carefully constructed problems in pure mathematics, the best-known being integer factorization.

Game theory equilibrium resolution

Regret-Based Nash Equilibrium Sorting Genetic Algorithm for Combinatorial Game Theory Problems with Multiple Players

We introduce a regret-based fitness assignment strategy for evolutionary algorithms to find Nash equilibria in noncooperative simultaneous combinatorial game theory problems where it is computationally intractable to enumerate all decision options of the players involved in the game. Applications of evolutionary algorithms to non-cooperative simultaneous games have been limited due to challenges in guiding the evolutionary search toward equilibria, which are usually inferior points in the objective space. We propose a regret-based approach to select candidate decision options of the players for the next generation in a multipopulation genetic algorithm called Regret-Based Nash Equilibrium Sorting Genetic Algorithm (RNESGA). We show that RNESGA can converge to multiple Nash equilibria in a single run using two- and three- player competitive knapsack games and other games from the literature. We also show that pure payoff-based fitness assignment strategies perform poorly in three-player games.

Genetic algorithms in economics

Genetic algorithms have increasingly been applied to economics since the pioneering work by John H. Miller in 1986. It has been used to characterize a variety of models including the cobweb model, the overlapping generations model, game theory, schedule optimization and asset pricing. Specifically, it has been used as a model to represent learning, rather than as a means for fitting a model.

Genetic algorithm in the cobweb model

The cobweb model is a simple supply and demand model for a good over *t* periods. Firms (agents) make a production quantity decision in a given period, however their output is not produced until the following period. Thus, the firms are going to have to use some sort of method to forecast what the future price will be. The GA is used as a sort of learning behaviour for the firms. Initially their quantity production decisions are random, however each period they learn a little more. The result is the agents converge within the area of the rational expectations (RATEX) equilibrium for the stable and unstable case. If the election operator is used, the GA converges exactly to the RATEX equilibrium.

There are two types of learning methods these agents can be deployed with: social learning and individual learning. In social learning, each firm is endowed with a single string which is used as its quantity production decision. It then compares this string against other firms' strings. In the individual learning case, agents are endowed with a pool of strings. These strings are then compared against other strings within the agent's population pool. This can be thought of as mutual competing ideas within a firm whereas in the social case, it can be thought of as a firm learning from more successful firms. Note that in the social case and in the individual learning case with identical cost functions, that this is a homogeneous solution, that is all agents' production decisions are identical. However, if the cost functions are not identical, this will result in a heterogeneous solution, where firms produce different quantities (note that they are still locally homogeneous, that is within the firm's own pool all the strings are identical).

After all agents have made a quantity production decision, the quantities are aggregated and plugged into a demand function to get a price. Each firm's profit is then calculated. Fitness values are then calculated as a function of profits. After the offspring pool is generated, hypothetical

fitness values are calculated. These hypothetical values are based on some sort of estimation of the price level, often just by taking the previous price level.

13. SUPERVISED LEARNING EXAMPLES

***A) Search the Internet**: Search the Internet for Examples from your specialty and present a brief description.*

What is machine learning in retail

Machine learning in retail involves the adoption of self-learning computer algorithms designed to process huge datasets, identify relevant metrics, recurring patterns, anomalies, or cause-effect relations among variables, and therefore get a deeper understanding of the dynamics driving this industry and the contexts where retailers operate. The more retail data machine learning systems process, the more they fine-tune their performance as they detect new correlations and better frame the business scenario they're analyzing.

Such capabilities are typically leveraged in two ways. First, machine learning can power augmented analytics solutions that, compared with traditional methods of statistical analysis, will peer way deeper into data, spot even the subtlest interconnections between data points, and better cope with new trends and ever-evolving phenomena.

Second, machine learning's pattern recognition paves the way for relevant developments in the AI field of so-called cognitive technologies, which allow machines to replicate some of the innate skills of human beings. This includes natural language processing systems leveraging machine learning to recognize the linguistic patterns of human communication to understand and mimic them, along with computer vision solutions tapping into algorithms to detect visual patterns and relate them to specific objects.

Business opportunities of machine learning in retail

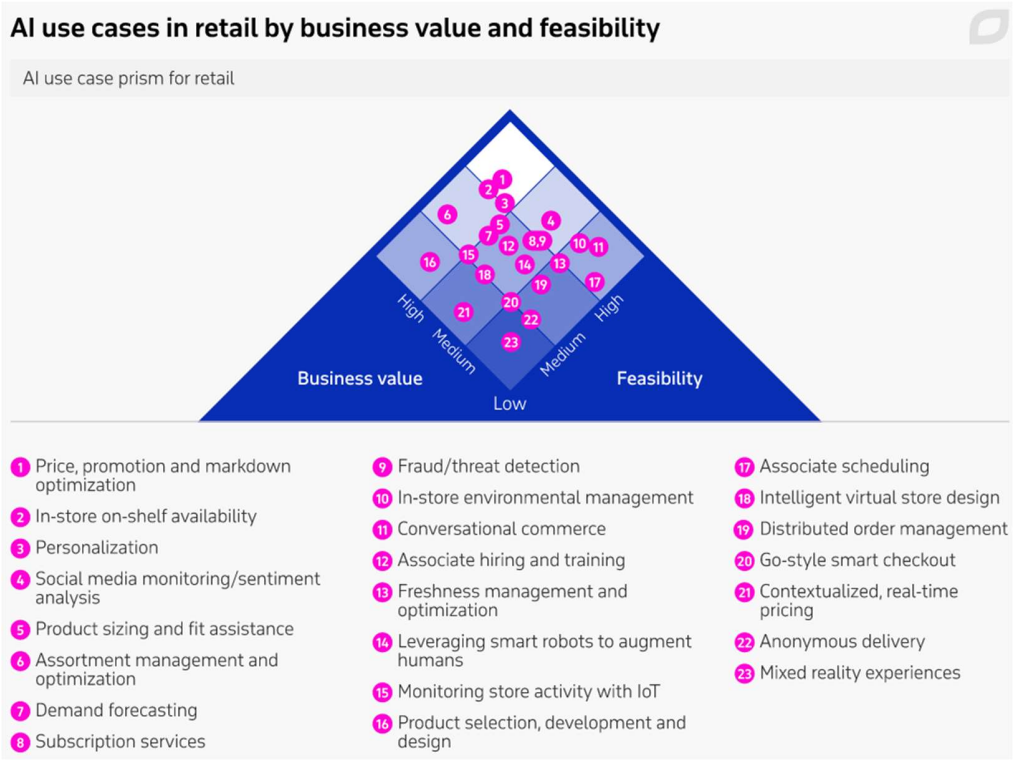
In practical terms, the above translates into the ability for retailers to implement machine learning and take advantage of it in a variety of business functions and scenarios, which include:

- **Market and customer analytics** to predict upcoming retail trends such as product demand fluctuations and set up suitable marketing, pricing and restocking strategies.
- **A fully personalized shopping experience** with recommendation engines, targeted advertising, dynamic pricing, and tailored promotions based on customers' needs.
- **Interactive solutions for digital stores** to replicate the typical in-store experience in a virtual scenario through chatbots, virtual assistants, and contextual shopping.
- **Machine learning-augmented logistics** to streamline product delivery via anticipatory shipping, smart route planning, and self-driving vehicles or drones.
- **Retail security measures** to protect your assets, staff, and customers with video surveillance and spot signs of fraud via machine learning-based anomaly detection.

These applications of machine learning in retail can be complemented with a full spectrum of additional use cases unlocked by artificial intelligence and all its sub-branches and varying both in business impact and feasibility, as detailed by Gartner in its 2021 *AI Use Case Prism for Retail*.

The state of the market

Nowadays, most AI-powered retail software solutions rely one way or another on machine learning algorithms' capabilities. As highlighted by Mordor Intelligence in its 2021 *Artificial Intelligence in Retail Market* report, in fact, machine learning represents one of the major drivers of the global AI market growth in this industry, as it allows market players to provide end-users with a more



personalized and interactive purchase experience.

Fortune Business Insight's 2020 study covering the same topic, on the other hand, pointed out machine learning's role in building forecasting models and offering valuable insights into ever-changing customer preferences. This makes machine learning the leading segment of the global AI market in retail, which seems set to grow from $3.75 billion in 2020 to $31.18 billion by 2028.

In this regard, it’s worth noting that the growing implementation of machine learning and related technologies in retail hasn't just been a reactive measure to better address the challenges mentioned in our introduction. Indeed, it has also represented a proactive initiative aimed at seizing the possibilities opened up by artificial intelligence, which ended up catalyzing general dynamics already undergone by the retail sector in previous years, such as the shift from a purely brick-and-mortar model to the coexistence of in-store shopping and ecommerce.

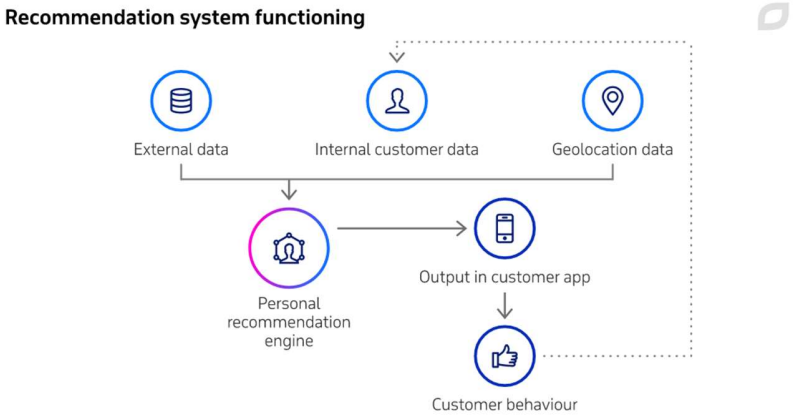
Let's explore these trends by better defining some of the AI use cases that harness the potential of machine learning in retail.

1. Recommendation engines

Since we've just mentioned the rapid transition from in-store to online shopping, we'll start our roundup from what we may consider the deus ex machina of major ecommerce platforms such as Amazon, namely machine learning-powered recommendation systems.

The role of these powerful engines in digital marketplaces represents the virtual reflection of a human sales assistants' duties in a physical shop, which is to understand what kind of customer we are dealing with and to connect them to the right product. Recommendation systems do so by:

- Segmenting users according to their personal data, including behavioral patterns and previous purchases on the platform, social media comments, browsing history, product reviews, and more.
- Complementing this analysis with product information and contextual parameters such as broader market trends or geographic location.
- Providing users with tailored suggestions matching their customer archetype and subsequently fine-tuning such recommendations based on user feedback, instead of letting them wander endlessly in a sprawling maze of products.



Data source: deloitte.com—The age of with Leveraging AI to connect the retail enterprise of the future, 2021

This process, however, can vary depending on the approach taken by each recommendation system:

- **Recommendation systems based on collaborative filtering** focus on users' perception of products and suggest merchandise already purchased by other customers with similar tastes.
- **Recommendation systems adopting content-based filtering** mostly consider the product's specific features and will offer items similar to those already bought by the customer in the past.

- **Hybrid systems** represent the fastest growing segment in the recommendation system market as they combine the mechanisms and pros of both approaches with proper techniques.

Scale up your sales with Itransition's machine learning solutions

Get in touch

2. Targeted marketing

Another tool leveraging the potential of machine learning in ecommerce but easily capable of driving in-store sales as well as targeted advertising. Its underlying mechanism is relatively similar to that behind recommendation systems. A machine learning-based predictive analytics system can gather and process user data from social media or ecommerce platforms to probe relevant metrics and shed light on their correlations. These may include:

- Behavioral factors, such as time on site, bounce rate and conversion rate
- Demographic variables, like age and gender
- Psychographic metrics relating to interests and personality
- Geographic data like user city and region

Based on such parameters and how they interact (such as the recurring relationship between age or cultural interests and favorite brands), the system will forecast customer interests and attitude towards certain products to target specific buyer personas with fully personalized content and ads. An example of successful targeted marketing comes from the American department store chain Macy's, which implemented a predictive analytics solution to fine-tune its email and website marketing campaigns, leading to a 10% increase in online sales in just three months after its introduction.

3. Contextual shopping

The opportunities for marketers unlocked by machine learning in retail don't end with targeted advertising. Another trick to shorten the virtual path connecting customers with the products they are looking for is contextual shopping. This highly interactive software solution, powered by machine learning algorithms and computer vision, can recognize and highlight the merchandise appearing in online content of major social media platforms, allowing users to reach your digital store and purchase the item they want with a simple click.

The video ecommerce platform provider AiBUY, for example, relied on Itransition's machine learning consulting and development services to create a similar tool. Take a look:

4. Chatbots and virtual shopping assistants

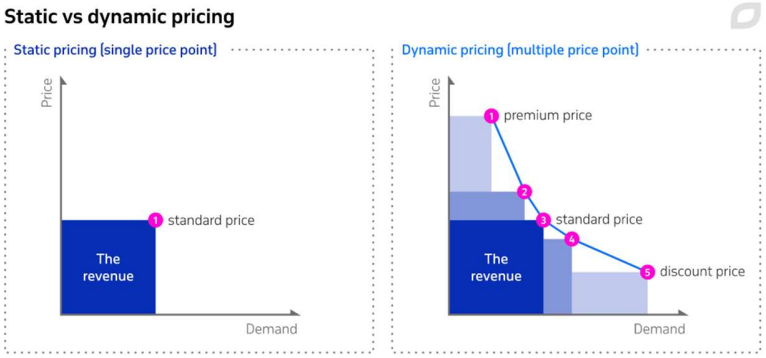
Chatbots, as contextual shopping, represent an embodiment of machine learning in retail focusing on interactivity. However, their capabilities come from a different cognitive technology, namely natural language processing, which allows them to assist customers 24/7 in a variety of tasks. This

may include sending notifications regarding new collections, helping users find the product they need, suggesting similar items based on recommendation engines' predictions, and so on.

Such versatile and tireless virtual assistants can be found all around the internet as they've already been deployed on the websites of several brands such as Burberry and Victoria's Secret. They've also been implemented on social media platforms like Facebook, which launched its Messenger bots in 2016, allowing businesses to partially automate their customer service operations.

5. Dynamic pricing

The clairvoyant capabilities of machine learning are not just about predicting which products customers would like to buy, but also how much they are willing to pay for them. In this regard, one of the techniques that has benefited most from implementing machine learning algorithms is dynamic pricing, which involves, as the name suggests, dynamic adjustments to the price of products based on changing circumstances.



To accurately predict the potential impact of such fluctuations on demand (the so-called price elasticity) and suggest price changes or promotions which may maximize profit while minimizing the risk of customer churn, machine learning systems can assess a wide spectrum of parameters which would be difficult to consider by performing this task manually. For example, they can crawl the web searching for prices of similar products offered by competitors, their hot deals and promotional initiatives in place, and the pricing history of particular products.

Machine learning algorithms can also take into account general market trends, product demand compared to supply, and the items with the lowest price in their respective product category, which typically results in a disproportionate share of demand. All these insights can be incredibly useful for setting a markdown pricing strategy and getting rid of obsolete merchandise at the end of a season.

Rely on Itransition's expertise to get value from your data

Let's talk

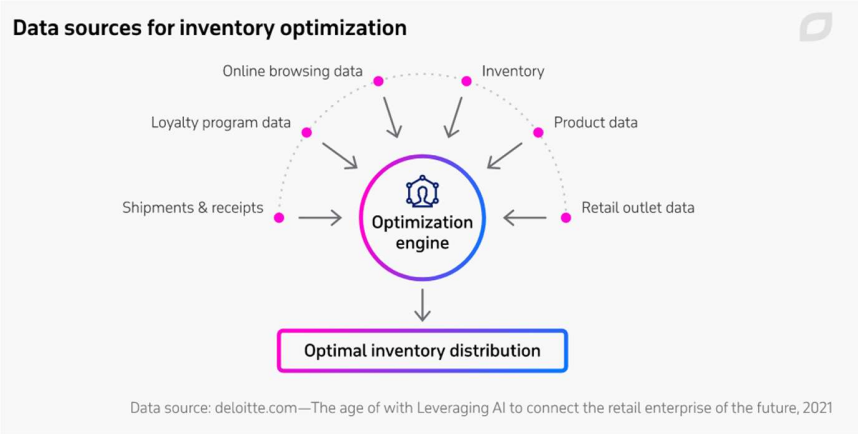
6. Demand prediction for inventory management

As we've just pointed out, prices obviously affect product demand and the adoption of machine learning in retail can help us figure out how this relationship works in order to optimize our pricing strategies. But things are more complicated than that, since demand trends on a large scale are driven by a vast range of variables:

- Business initiatives such as the aforementioned pricing decisions, promotions, and the product assortment and display inside a store.
- Purchase patterns relating to weekdays, seasonality, or celebrations (think about chocolate hearts for Valentine's Day).
- External motivators that encourage customers to leave home, like good weather or local events.

Monitoring such factors to predict fluctuations in product demand is an essential component of proper inventory and supply chain management, as it allows retailers to keep stocks replenished. In this regard, Deloitte highlighted that stockouts may cost retailers up to $1 trillion a year in lost sales and reported that a major German sportswear retailer successfully leveraged machine learning algorithms to streamline restocking operations in its online and in-store branches.

To keep track of all these parameters, however, retailers should fuel machine learning-based predictive analytics systems with a constant stream of information coming from multiple data channels, which include "virtual sources" like social media and ecommerce platforms, along with data collected from physical stores. These may include "virtual sources" such as social media, ecommerce platforms, and the integration of voice commerce along with data collected from physical stores.



Besides traditional data sources, it's worth mentioning new, more unconventional ways of harvesting data to enhance inventory management. A striking example comes from Amazon Go, a chain of convenience stores relying on machine learning-powered cameras and computer vision to update product stock data in real time based on the products customers grab off the shelves. Furthermore, these computer vision systems are able to:

- Monitor customers' behavior, including walking and gazing patterns, to help retailers understand people's interests and restructure stores accordingly.
- Identify the most popular merchandise for each buyer archetype. For example, if elderly people tend to frequent the store on weekdays rather than weekends, then the products specific to this age group can be promoted during those days.
- Suggest specific marketing actions to store managers. For example, if a particular brand of milk is selling better than the competition, it can be positioned more prominently.

- Turn stores into smart, cashier-free points of sale where fully automated camera-based checkout replaces long queues at the cash desks.

7. Delivery optimization

Products come and go and optimizing how they leave your store is as important as perfecting their replenishment. Machine learning-powered route planning represents a major use case of AI in transportation as it leverages sensors, cameras and other devices interconnected through the Internet of things to collect real-time weather and traffic data and calculate the fastest route for deliveries.

Anheuser-Busch, for example, adopted machine learning for route optimization in a pilot program covering two American cities and aimed at speeding up daily deliveries. The algorithms considered factors such as weather conditions and the driver's experience to suggest the best delivery time for each consumer. After a few months of running the pilot program, Anheuser-Busch reviewed the success metrics, including driver satisfaction and working hours, and decided to extend this navigation approach to all of its US wholesalers.

But what if we could reduce delivery times by sending products even before customers hit the "buy" button? This is the approach adopted by Amazon, which implemented an "anticipatory shipping" system based on machine learning and able to predict what its users are likely to order in the future based on their previous purchases, wishlist, and other parameters. This solution enables Amazon to send merchandise to a closer hub while waiting for the customer to actually order it and then dispatch it via standard shipping, which is significantly less expensive than same-day delivery.

8. Self-driving vehicles

Optimizing delivery routes doesn't change the fact that drivers still get tired and can't operate a vehicle 24 hours a day. Robots, instead, don't need a nap and can work around the clock. That's the idea behind deploying self-driving vehicles powered by machine learning and computer vision for logistics and retail.

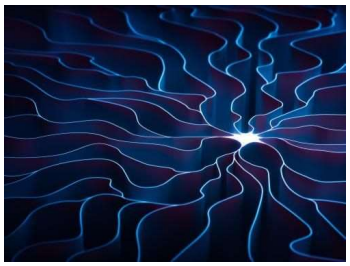
Different retailers are already experimenting with self-driving vehicles for their last-mile deliveries. Since 2019, Amazon has been testing Amazon Scout, a self-driving electric device the size of a cooler. Amazon started with six of these devices delivering packages on weekdays in daylight and operating under human supervision to ensure they safely navigated around pets and pedestrians.

Kroger, on the other hand, partnered with Google-backed robotics corporation Nuro to deliver groceries in self-driving vehicles seven days a week. The company just recently launched the 3rd-generation of its flagship model, designed to double the cargo volume and handle more deliveries.

However, this is only the beginning, as autonomous vehicles may soon take to the skies. Once again, Amazon is one of the pioneering operators in this field, as in the future it may be able to deliver packages in just 30 minutes with its "Prime Air" drone-based shipping system. On the other

hand, the project is still far from gaining full momentum, as several incidents occurred over the course of a four-month test period, as reported by Bloomberg.

Get your custom machine learning solution from Itransition



Machine learning consulting

We offer machine learning consulting, modeling and implementation services, helping retailers with different business goals, needs, and budgets apply this innovative technology to their advantage.

9. Video surveillance

Having a Big Brother watching you can be the lesser evil when it comes to ensuring a safe retail environment. Video surveillance solutions harnessing the capabilities of machine learning and computer visions for retail can spot any other suspicious behavioral patterns which are common among thieves, for example, by detecting if someone picked an item and hid it in their pocket or backpack.

If retailers expect to reliably identify theft with computer vision, however, it will take more than "just" high-resolution cameras and algorithms, since they'll also need a huge amount of labeled examples of visual data representing distinct patterns of theft behavior. For example, the Japanese technology startup Vaak developed AI software designed to catch shoplifters in action and alert store managers. The CEO of Vaak admitted to training machine learning algorithms on 100,000 hours of video data depicting over 100 behavioral aspects, including hand movements, facial expressions, walking patterns, etc.

10. Fraud detection

In a world driven by technological progress and complex socioeconomic dynamics towards full digitalization, products disappearing from the shelves are neither the only nor the worst threat to retailers and their customers. As pointed out by the 2020 *Nilson report*, electronic payment fraud is a growing phenomenon affecting several industries and generating global losses of $28.65 billion in 2019 alone.

That's why many corporations, especially those operating in ecommerce, rely on machine learning for fraud detection. Indeed, algorithms can easily spot suspicious account operations breaking standard behavioral patterns, such as a spike in transaction frequency or money transfers with other high-risk accounts.

Machine learning systems also represent a powerful weapon to combat identity theft by monitoring users' purchase habits and transaction data. Furthermore, they can implement computer vision features to scan personal documents and verify user identity via face recognition and biometrics.

Lots of enthusiasm and a pinch of caution

AI and machine learning are swiftly invading the retail space, opening new opportunities for retailers to differentiate themselves from their competitors, including:

- Retaining current customers and acquiring new ones thanks to personalization
- Saving time and human effort by optimizing delivery routes
- Saving money through effective inventory management and theft prevention
- Gaining a reputation as an innovative and committed retailer who values its customers and goes the extra mile to please them

However, as amazing as it is, machine learning is still "just" a tool, and using it wisely is up to us. As highlighted by McKinsey in its 2021 *The dos and don'ts of dynamic pricing in retail* article, during the pandemic, several retailers massively hiked cleaning products' prices due to an apparent demand spike monitored by dynamic-pricing algorithms. The result was a significant backslash from customers which perceived their decision as an opportunistic move.

The first lesson we can draw from this episode is that people have an uncontrollable desire to clean their apartment when social distancing measures keep them stuck at home. The second, and most important one, is that proper human supervision of machine learning-driven strategies, at least for now, may be a necessity rather than a choice and should be performed with reason in mind.

https://www.itransition.com/machine-learning/retail

14. EXAMPLES OF UNSUPERVISED LEARNING

A) Search the Internet*: Search the Internet for Examples from your specialty and present a brief description.*

Answered totally in question 13

15. EXAMPLES OF REINFORCEMENT LEARNING

A) Search the Internet*: Search the Internet for Examples from your specialty and present a brief description.*

Answered totally in question 13

16. FINDING/UNDERSTANDING MATLAB/OCTAVE TOOLS ONLINE GUIDES FOR FINDING POLYNOMIAL COEFFICIENTS

A) Search the Internet*: instructions and examples for the MatLab/Octave polynomial identification tools.*

28.5 Polynomial Interpolation

Octave comes with good support for various kinds of interpolation, most of which are described in Interpolation. One simple alternative to the functions described in the aforementioned chapter, is to fit a single polynomial, or a piecewise polynomial (spline) to some given data points. To avoid

a highly fluctuating polynomial, one most often wants to fit a low-order polynomial to data. This usually means that it is necessary to fit the polynomial in a least-squares sense, which just is what the polyfit function does.

p = **polyfit** *(x, y, n)*
[p, s] = **polyfit** *(x, y, n)*
[p, s, mu] = **polyfit** *(x, y, n)*

In situations where a single polynomial isn't good enough, a solution is to use several polynomials pieced together. The function `splinefit` fits a piecewise polynomial (spline) to a set of data.

pp = **splinefit** *(x, y, breaks)*
pp = **splinefit** *(x, y, p)*
pp = **splinefit** *(..., "periodic", periodic)*
pp = **splinefit** *(..., "robust", robust)*
pp = **splinefit** *(..., "beta", beta)*
pp = **splinefit** *(..., "order", order)*
pp = **splinefit** *(..., "constraints", constraints)*

Fit a piecewise cubic spline with breaks (knots) *breaks* to the noisy data, *x* and *y*.

x is a vector, and *y* is a vector or N-D array. If *y* is an N-D array, then *x*(j) is matched to *y*(:,…,:,j).

p is a positive integer defining the number of intervals along *x*, and *p*+1 is the number of breaks. The number of points in each interval differ by no more than 1.

The optional property *periodic* is a logical value which specifies whether a periodic boundary condition is applied to the spline. The length of the period is `max (breaks) - min (breaks)`. The default value is `false`.

The optional property *robust* is a logical value which specifies if robust fitting is to be applied to reduce the influence of outlying data points. Three iterations of weighted least squares are performed. Weights are computed from previous residuals. The sensitivity of outlier identification is controlled by the property *beta*. The value of *beta* is restricted to the range, 0 < *beta* < 1. The default value is *beta* = 1/2. Values close to 0 give all data equal weighting. Increasing values of *beta* reduce the influence of outlying data. Values close to unity may cause instability or rank deficiency.

The fitted spline is returned as a piecewise polynomial, *pp*, and may be evaluated using `ppval`.

The splines are constructed of polynomials with degree *order*. The default is a cubic, *order*=3. A spline with P pieces has P+*order* degrees of freedom. With periodic boundary conditions the degrees of freedom are reduced to P.

The optional property, *constraints*, is a structure specifying linear constraints on the fit. The structure has three fields, `"xc"`, `"yc"`, and `"cc"`.

`"xc"`

Vector of the x-locations of the constraints.

`"yc"`

Constraining values at the locations *xc*. The default is an array of zeros.

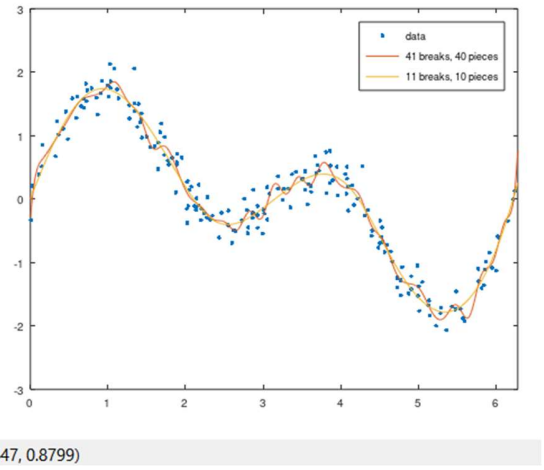
`"cc"`

Coefficients (matrix). The default is an array of ones. The number of rows is limited to the order of the piecewise polynomials, *order*.

Constraints are linear combinations of derivatives of order 0 to *order*-1 according to `cc(1,j) * y(xc(j)) + cc(2,j) * y'(xc(j)) + ... = yc(:,...,:,j).`

The number of *breaks* (or knots) used to construct the piecewise polynomial is a significant factor in suppressing the noise present in the input data, *x* and *y*. This is demonstrated by the example below.

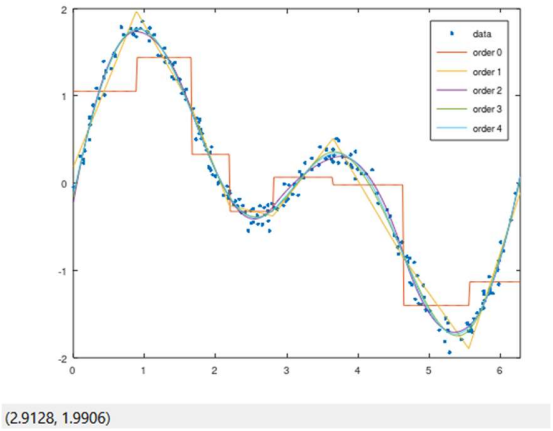
```
x = 2 * pi * rand (1, 200);
y = sin (x) + sin (2 * x) + 0.2 * randn (size
(x));
## Uniform breaks
breaks = linspace (0, 2 * pi, 41); % 41 breaks,
40 pieces
pp1 = splinefit (x, y, breaks);
## Breaks interpolated from data
pp2 = splinefit (x, y, 10);  % 11 breaks, 10
pieces
## Plot
xx = linspace (0, 2 * pi, 400);
y1 = ppval (pp1, xx);
y2 = ppval (pp2, xx);
plot (x, y, ".", xx, [y1; y2])
axis tight
ylim auto
legend ({"data", "41 breaks, 40 pieces", "11
breaks, 10 pieces"})
```



Comparison of a fitting a piecewise polynomial with 41 breaks to one with 11 breaks. The fit with the large number of breaks exhibits a fast ripple that is not present in the underlying function.

The piecewise polynomial fit, provided by `splinefit`, has continuous derivatives up to the *order*-1. For example, a cubic fit has continuous first and second derivatives. This is demonstrated by the code

```
## Data (200 points)
x = 2 * pi * rand (1, 200);
y = sin (x) + sin (2 * x) + 0.1 * randn (size (x));
## Piecewise constant
pp1 = splinefit (x, y, 8, "order", 0);
## Piecewise linear
pp2 = splinefit (x, y, 8, "order", 1);
## Piecewise quadratic
pp3 = splinefit (x, y, 8, "order", 2);
## Piecewise cubic
pp4 = splinefit (x, y, 8, "order", 3);
## Piecewise quartic
pp5 = splinefit (x, y, 8, "order", 4);
## Plot
xx = linspace (0, 2 * pi, 400);
y1 = ppval (pp1, xx);
y2 = ppval (pp2, xx);
y3 = ppval (pp3, xx);
y4 = ppval (pp4, xx);
```

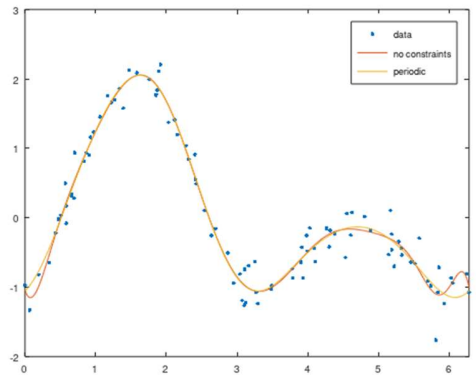


```
y5 = ppval (pp5, xx);
plot (x, y, ".", xx, [y1; y2; y3; y4; y5])
axis tight
ylim auto
legend ({"data", "order 0", "order 1", "order 2", "order 3", "order 4"})
```

Comparison of a piecewise constant, linear, quadratic, cubic, and quartic polynomials with 8 breaks to noisy data. The higher order solutions more accurately represent the underlying function, but come with the expense of computational complexity.

When the underlying function to provide a fit to is periodic, `splinefit` is able to apply the boundary conditions needed to manifest a periodic fit. This is demonstrated by the code below.

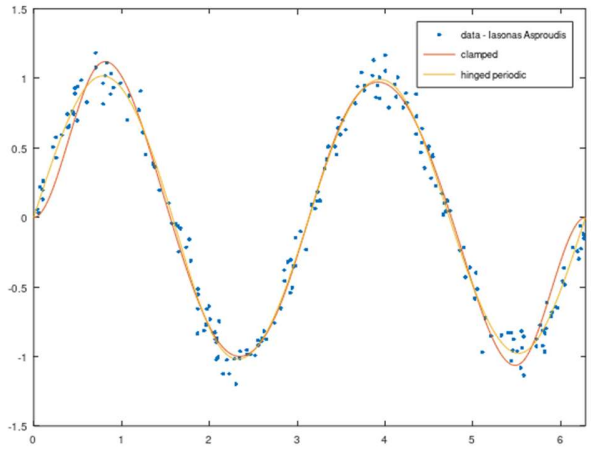
```
## Data (100 points)
x = 2 * pi * [0, (rand (1, 98)), 1];
y = sin (x) - cos (2 * x) + 0.2 * randn (size
(x));
## No constraints
pp1 = splinefit (x, y, 10, "order", 5);
## Periodic boundaries
pp2 = splinefit (x, y, 10, "order", 5, "periodic",
true);
## Plot
xx = linspace (0, 2 * pi, 400);
y1 = ppval (pp1, xx);
y2 = ppval (pp2, xx);
plot (x, y, ".", xx, [y1; y2])
axis tight
ylim auto
legend ({"data", "no constraints", "periodic"})
```



Comparison of piecewise polynomial fits to a noisy periodic function with, and without, periodic boundary conditions.

More complex constraints may be added as well. For example, the code below illustrates a periodic fit with values that have been clamped at the endpoints, and a second periodic fit which is hinged at the endpoints.

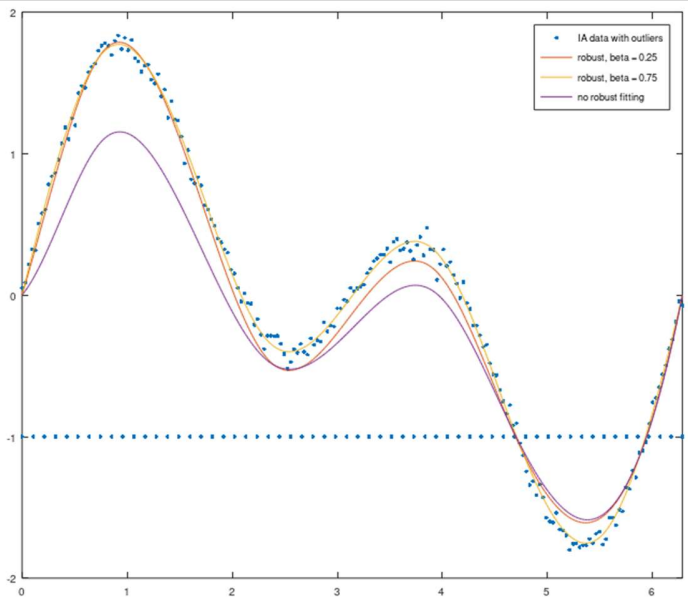
```
## Data (200 points)
x = 2 * pi * rand (1, 200);
y = sin (2 * x) + 0.1 * randn (size (x));
## Breaks
breaks = linspace (0, 2 * pi, 10);
## Clamped endpoints, y = y' = 0
xc = [0, 0, 2*pi, 2*pi];
cc = [(eye (2)), (eye (2))];
con = struct ("xc", xc, "cc", cc);
pp1 = splinefit (x, y, breaks,
"constraints", con);
## Hinged periodic endpoints, y = 0
con = struct ("xc", 0);
pp2 = splinefit (x, y, breaks,
"constraints", con, "periodic", true);
## Plot
xx = linspace (0, 2 * pi, 400);
y1 = ppval (pp1, xx);
y2 = ppval (pp2, xx);
plot (x, y, ".", xx, [y1; y2])
axis tight
ylim auto
legend ({"data", "clamped", "hinged periodic"})
```



Comparison of two periodic piecewise cubic fits to a noisy periodic signal. One fit has its endpoints clamped and the second has its endpoints hinged.

The `splinefit` function also provides the convenience of a *robust* fitting, where the effect of outlying data is reduced. In the example below, three different fits are provided. Two with differing levels of outlier suppression and a third illustrating the non-robust solution.

```
## Data
x = linspace (0, 2*pi, 200);
y = sin (x) + sin (2 * x) + 0.05 * randn (size (x));
## Add outliers
x = [x, linspace(0,2*pi,60)];
y = [y, -ones(1,60)];
## Fit splines with hinged conditions
con = struct ("xc", [0, 2*pi]);
## Robust fitting, beta = 0.25
pp1 = splinefit (x, y, 8, "constraints",
con, "beta", 0.25);
## Robust fitting, beta = 0.75
pp2 = splinefit (x, y, 8, "constraints",
con, "beta", 0.75);
## No robust fitting
pp3 = splinefit (x, y, 8, "constraints",
con);
## Plot
xx = linspace (0, 2*pi, 400);
y1 = ppval (pp1, xx);
y2 = ppval (pp2, xx);
y3 = ppval (pp3, xx);
plot (x, y, ".", xx, [y1; y2; y3])
legend ({"data with outliers","robust,
beta = 0.25", ...
         "robust, beta = 0.75", "no
robust fitting"})
axis tight
ylim auto
```



Comparison of two different levels of robust fitting (*beta* = 0.25 and 0.75) to noisy data combined with outlying data. A conventional fit, without robust fitting (*beta* = 0) is also included.

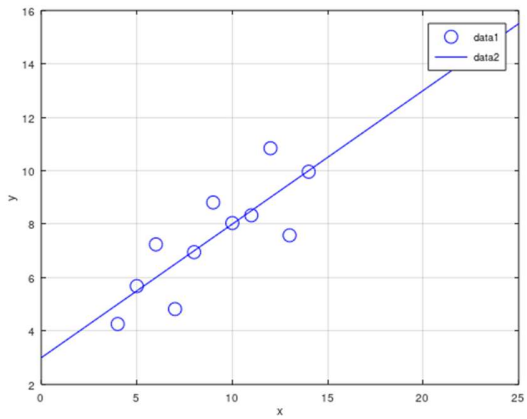
17. FINDING AND TESTING ANSCOMBE'S QUARTET

***A) Search the Internet for**: Anscombe's 4 data sets and try modeling them with linear, quadratic, or other models.*

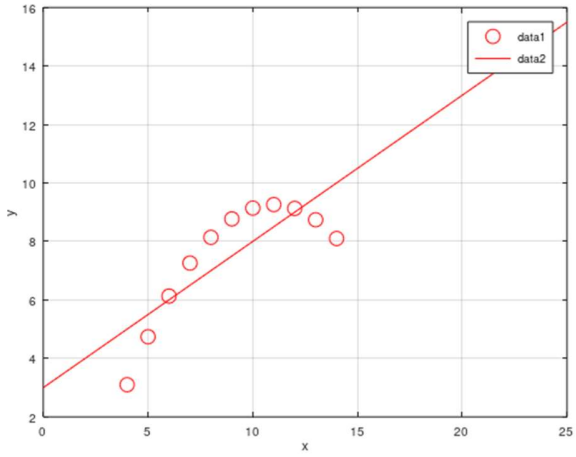
https://gist.github.com/ericbusboom/b2ac1d366c005cd2ed8c

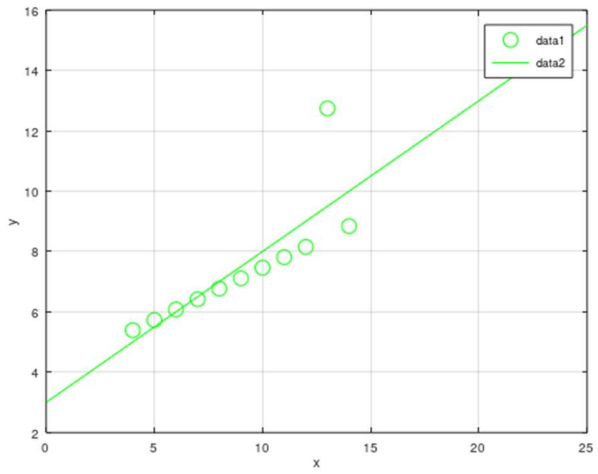
Anscombe_Quartet_linear.m

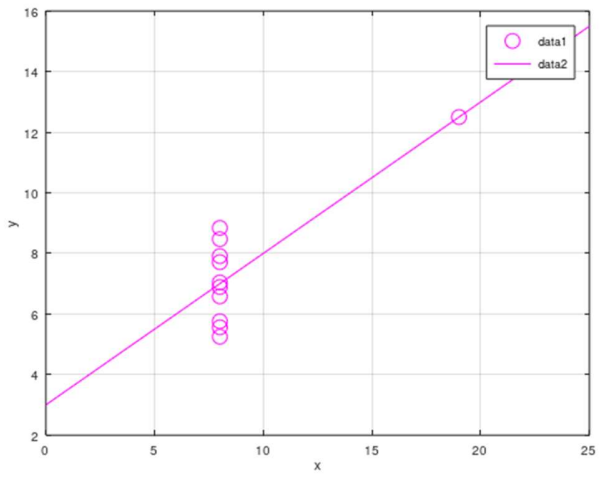
```
clear all; clc;

x1=[10 8 13 9 11 14 6 4 12 7 5];
x2=x1;
x3=x1;
x4=[8 8 8 8 8 8 8 19 8 8 8];

y1=[8.04 6.95 7.58 8.81 8.33 9.96 7.24 4.26 10.84 4.82 5.68];
y2=[9.14 8.14 8.74 8.77 9.26 8.1 6.13 3.1 9.13 7.26 4.74];
y3=[7.46 6.77 12.74 7.11 7.81 8.84 6.08 5.39 8.15 6.42 5.73];
y4=[6.58 5.76 7.71 8.84 8.47 7.04 5.25 12.5 5.56 7.91 6.89];

%% Linear Regression (least squares method)
%category I

X1 = [ones(size(x1)); x1];
```



```
Y1 = y1;

theta = Y1*pinv(X1);

f = @(x1) theta(1) + theta(2)*x1;

xx1 = linspace(0,25,100);
yy1 = f(xx1);

figure;
plot(x1,y1,'bo'); hold on;
plot(xx1,yy1,'b'); hold on;
xlabel('x');
ylabel('y');
grid on;
legend show;

% category II

X2 = [ones(size(x2)); x2];
Y2 = y2;

theta = Y2*pinv(X2);

f = @(x2) theta(1) + theta(2)*x2;

xx2 = linspace(0,25,100);
yy2 = f(xx2);

figure;
plot(x2,y2,'ro'); hold on;
plot(xx2,yy2,'r'); hold on;
xlabel('x');
ylabel('y');
grid on;
legend show;

%category III

X3 = [ones(size(x3)); x3];
Y3 = y3;

theta = Y3*pinv(X3);

f = @(x3) theta(1) + theta(2)*x3;

xx3 = linspace(0,25,100);
yy3 = f(xx3);

figure;
plot(x3,y3,'go'); hold on;
plot(xx3,yy3,'g'); hold on;
xlabel('x');
ylabel('y');
grid on;
legend show;

%category IV

X4 = [ones(size(x4)); x4];
Y4 = y4;

theta = Y4*pinv(X4);

f = @(x4) theta(1) + theta(2)*x4;
```



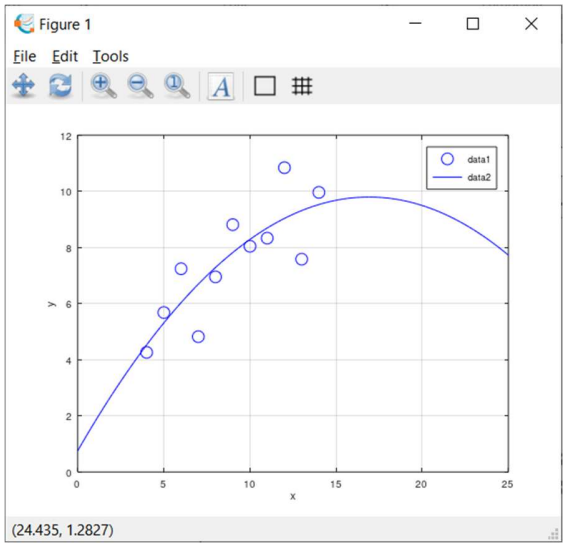


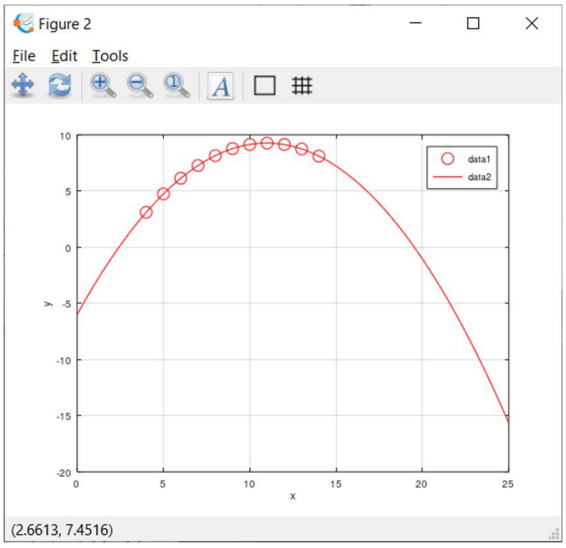


```
xx4 = linspace(0,25,100);
yy4 = f(xx4);

figure;
plot(x4,y4,'mo'); hold on;
plot(xx4,yy4,'m',); hold on;
xlabel('x');
ylabel('y');
grid on;
legend show;
```

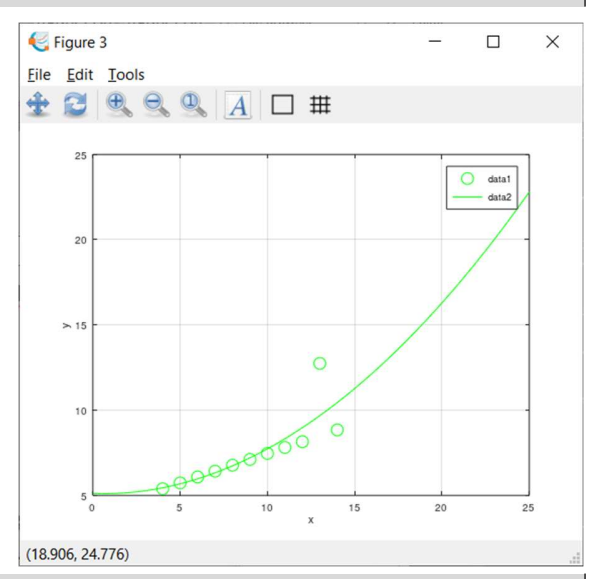
Anscombe_Quartet_non-linear.m

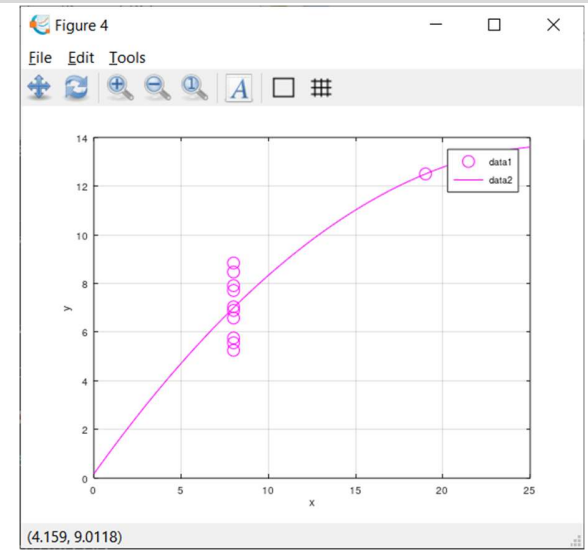
```
clear all; clc;

x1=[10 8 13 9 11 14 6 4 12 7 5];
x2=x1;
x3=x1;
x4=[8 8 8 8 8 8 8 19 8 8 8];

y1=[8.04 6.95 7.58 8.81 8.33 9.96 7.24 4.26 10.84 4.82 5.68];
y2=[9.14 8.14 8.74 8.77 9.26 8.1 6.13 3.1 9.13 7.26 4.74];
y3=[7.46 6.77 12.74 7.11 7.81 8.84 6.08 5.39 8.15 6.42 5.73];
y4=[6.58 5.76 7.71 8.84 8.47 7.04 5.25 12.5 5.56 7.91 6.89];

%%Non-Linear Regression

% category I

X1 = [ones(size(x1));
    x1
    x1.^2];

Y1 = y1;

theta = Y1*pinv(X1);

f = @(x1) theta(1) + theta(2)*x1 + theta(3)*x1.^2;

xx1 = linspace(0,25,100);
yy1 = f(xx1);

figure;
plot(x1,y1,'bo'); hold on;
plot(xx1,yy1,'b'); hold on;
xlabel('x');
ylabel('y');
grid on;
legend show;

% category II

X2 = [ones(size(x2));
    x2
    x2.^2];

Y2 = y2;

theta = Y2*pinv(X1);

f = @(x2) theta(1) + theta(2)*x2 + theta(3)*x2.^2;

xx2 = linspace(0,25,100);
yy2 = f(xx2);
```





```
figure;
plot(x2,y2,'ro'); hold on;
plot(xx2,yy2,'r'); hold on;
xlabel('x');
ylabel('y');
grid on;
legend show;

% category III

X3 = [ones(size(x3));
    x3
    x3.^2];

Y3 = y3;

theta = Y3*pinv(X3);

f = @(x3) theta(1) + theta(2)*x3 + theta(3)*x3.^2;

xx3 = linspace(0,25,100);
yy3 = f(xx3);

figure;
plot(x3,y3,'go'); hold on;
plot(xx3,yy3,'g'); hold on;
xlabel('x');
ylabel('y');
grid on;
legend show;

% category IV

X4 = [ones(size(x4));
    x4
    x4.^2];

Y4 = y4;

theta = Y4*pinv(X4);

f = @(x4) theta(1) + theta(2)*x4 + theta(3)*x4.^2;

xx4 = linspace(0,25,100);
yy4 = f(xx4);

figure;
plot(x4,y4,'mo'); hold on;
plot(xx4,yy4,'m'); hold on;
xlabel('x');
ylabel('y');
grid on;
legend show;
```





18. SEARCH FOR MODEL VALIDATION CRITERIA AND METHODS

***A) Search the Internet for**: VALIDATION methods and describe their operation*

What are methods used for validating models?

Cross-validation (CV), leave-one-out, or k-fold is the frequently used method for model validation. If validation with an independent dataset is not possible due to the small sample size, CV is very economical. However, CV has been shown to yield an overoptimistic estimate of prediction ability.

Cross-validation (statistics)

Cross-validation,[2][3][4] sometimes called **rotation estimation**[5][6][7] or **out-of-sample testing**, is any of various similar model validation techniques for assessing how the results of a statistical analysis will generalize to an independent data set. Cross-validation is a resampling method that uses different portions of the data to test and train a model on different iterations. It is mainly used in settings where the goal is prediction, and one wants to estimate how accurately a predictive model will perform in practice. In a prediction problem, a model is usually given a dataset of *known data* on which training is run (*training dataset*), and a dataset of *unknown data* (or *first seen* data) against which the model is tested (called the validation dataset or *testing set*).[8][9] The goal of cross-validation is to test the model's ability to predict new data that was not used in estimating it, in order to flag problems like overfitting or selection bias[10] and to give an insight on how the model will generalize to an independent dataset (i.e., an unknown dataset, for instance from a real problem).
One round of cross-validation involves partitioning a sample of data into complementary subsets, performing the analysis on one subset (called the *training set*), and validating the analysis on the other subset (called the *validation set* or *testing set*). To reduce variability, in most methods multiple rounds of cross-validation are performed using different partitions, and the validation results are combined (e.g. averaged) over the rounds to give an estimate of the model's predictive performance.
In summary, cross-validation combines (averages) measures of fitness in prediction to derive a more accurate estimate of model prediction performance.

Leave-one-out Validation Method

Definition. Leave-one-out cross-validation is **a special case of cross-validation where the number of folds equals the number of instances in the data set**. Thus, the learning algorithm is applied once for each instance, using all other instances as a training set and using the selected instance as a single-item test set ...

K-fold cross-validation method

K-fold Cross-Validation is **when the dataset is split into a K number of folds and is used to evaluate the model's ability when given new data**. K refers to the number of groups the data sample is split into. For example, if you see that the k-value is 5, we can call this a 5-fold cross-validation.

***B) Search the Internet**: Model criteria and describe how they are calculated.*

Akaike Information Criterion | When & How to Use It (Example)
Published on March 26, 2020 by Rebecca Bevans. Revised on November 18, 2022.
The **Akaike information criterion (AIC)** is a mathematical method for evaluating how well a model fits the data it was generated from. In statistics, AIC is used to compare different possible models and determine which one is the best fit for the data. AIC is calculated from:

- the number of independent variables used to build the model.
- the maximum likelihood estimate of the model (how well the model reproduces the data).

The best-fit model according to AIC is the one that explains the greatest amount of variation using the fewest possible independent variables.
Akaike information criterion exampleYou want to know whether drinking sugar-sweetened beverages influences body weight. You have collected secondary data from a national health survey that contains observations on sugar-sweetened beverage consumption, age, sex, and BMI (body mass index).
To find out which of these variables are important for predicting the relationship between sugar-sweetened beverage consumption and body weight, you create several possible models and compare them using AIC.

When to use AIC
In statistics, AIC is most often used for model selection. By calculating and comparing the AIC scores of several possible models, you can choose the one that is the best fit for the data.
When testing a hypothesis, you might gather data on variables that you aren't certain about, especially if you are exploring a new idea. You want to know which of the independent variables you have measured explain the variation in your dependent variable.
A good way to find out is to create a set of models, each containing a different combination of the independent variables you have measured. These combinations should be based on:

- Your knowledge of the study system – avoid using parameters that are not logically connected, since you can find spurious correlations between almost anything!
- Your experimental design – for example, if you have split two treatments up among test subjects, then there is probably no reason to test for an interaction between the two treatments.

Once you've created several possible models, you can use AIC to compare them. Lower AIC scores are better, and AIC penalizes models that use more parameters. So if two models explain the same amount of variation, the one with fewer parameters will have a lower AIC score and will be the better-fit model.
Model selection exampleIn a study of how hours spent studying and test format (multiple choice vs. written answers) affect test scores, you create two models:

1. Final test score in response to hours spent studying
2. Final test score in response to hours spent studying + test format

You find an r^2 of 0.45 with a *p* value less than 0.05 for model 1, and an r^2 of 0.46 with a *p* value less than 0.05 for model 2. Model 2 fits the data slightly better – but was it worth it to add another parameter just to get this small increase in model fit?

You run an AIC test to find out, which shows that model 1 has the lower AIC score because it requires less information to predict with almost the exact same level of precision. Another way to think of this is that the increased precision in model 2 could have happened by chance.
From the AIC test, you decide that model 1 is the best model for your study.

Bayesian information criterion

The Bayesian information criterion (BIC) is one of the most widely known and pervasively used tools in statistical model selection. Its popularity is derived from its computational simplicity and effective performance in many modeling frameworks, including Bayesian applications where prior distributions may be elusive. The criterion was derived by Schwarz (*Ann Stat* 1978, 6:461–464) to serve as an asymptotic approximation to a transformation of the Bayesian posterior probability of a candidate model. This article reviews the conceptual and theoretical foundations for BIC, and also discusses its properties and applications. *WIREs Comput Stat* 2012, 4:199–203. doi: 10.1002/wics.199
This article is categorized under:

- Statistical and Graphical Methods of Data Analysis > Bayesian Methods and Theory
- Statistical and Graphical Methods of Data Analysis > Information Theoretic Methods
- Statistical Learning and Exploratory Methods of the Data Sciences > Modeling Methods

19-22 WORK – PROJECT

MODELING AND FORECASTING THE SPREAD OF AN EPIDEMIC/PANDEMIC

Produce a model of the spread of the Covid-19 epidemic/pandemic from the daily data you will find on the site:

https://data.europa.eu/data/datasets/covid-19-coronavirus-data?locale=en

Download the first file "COVID-19 cases worldwide" rename it to covid19.csv, and after extracting the data for Greece (covic19_GR.csv), use it to find a polynomial linear regression model that describes it satisfactorily and then predict pandemic prices for the next 3-month.

https://www.ecdc.europa.eu/en/publications-data/download-historical-data-20-june-2022-weekly-number-new-reported-covid-19-cases

➔ C:\Users\jasproudis\Downloads\data.csv

1	country	country_code	continent	population	indicator	weekly_count	year_week	rate_14_day	cumulative_count	source
12822	Czechia	CZE	Europe	10701777	cases	0	2020-01	NA	0	TESSy COVID-19
12823	Czechia	CZE	Europe	10701777	cases	0	2020-02	0	0	TESSy COVID-19
12824	Czechia	CZE	Europe	10701777	cases	0	2020-03	0	0	TESSy COVID-19
12825	Czechia	CZE	Europe	10701777	cases	0	2020-04	0	0	TESSy COVID-19
12826	Czechia	CZE	Europe	10701777	cases	0	2020-05	0	0	TESSy COVID-19
12827	Czechia	CZE	Europe	10701777	cases	0	2020-06	0	0	TESSy COVID-19
12828	Czechia	CZE	Europe	10701777	cases	0	2020-07	0	0	TESSy COVID-19
12829	Czechia	CZE	Europe	10701777	cases	0	2020-08	0	0	TESSy COVID-19
12830	Czechia	CZE	Europe	10701777	cases	3	2020-09	0.028032728	3	TESSy COVID-19
12831	Czechia	CZE	Europe	10701777	cases	30	2020-10	0.308360004	33	TESSy COVID-19
12832	Czechia	CZE	Europe	10701777	cases	266	2020-11	2.765895795	299	TESSy COVID-19
12833	Czechia	CZE	Europe	10701777	cases	866	2020-12	10.57768257	1165	TESSy COVID-19
12834	Czechia	CZE	Europe	10701777	cases	1661	2020-13	23.61290092	2826	TESSy COVID-19
12835	Czechia	CZE	Europe	10701777	cases	1773	2020-14	32.08812892	4599	TESSy COVID-19
12836	Czechia	CZE	Europe	10701777	cases	1405	2020-15	29.69600282	6004	TESSy COVID-19
12837	Czechia	CZE	Europe	10701777	cases	755	2020-16	20.18356391	6759	TESSy COVID-19
12838	Czechia	CZE	Europe	10701777	cases	658	2020-17	13.20341472	7417	TESSy COVID-19
12839	Czechia	CZE	Europe	10701777	cases	379	2020-18	9.689979524	7796	TESSy COVID-19
12840	Czechia	CZE	Europe	10701777	cases	344	2020-19	6.755887363	8140	TESSy COVID-19
12841	Czechia	CZE	Europe	10701777	cases	359	2020-20	6.569002512	8499	TESSy COVID-19
12842	Czechia	CZE	Europe	10701777	cases	486	2020-21	7.895884954	8985	TESSy COVID-19
12843	Czechia	CZE	Europe	10701777	cases	318	2020-22	7.51277101	9303	TESSy COVID-19
12844	Czechia	CZE	Europe	10701777	cases	367	2020-23	6.400806146	9670	TESSy COVID-19
12845	Czechia	CZE	Europe	10701777	cases	399	2020-24	7.157689793	10069	TESSy COVID-19
12846	Czechia	CZE	Europe	10701777	cases	475	2020-25	8.166867988	10544	TESSy COVID-19
12847	Czechia	CZE	Europe	10701777	cases	1106	2020-26	14.77324747	11650	TESSy COVID-19
12848	Czechia	CZE	Europe	10701777	cases	910	2020-27	18.83799298	12560	TESSy COVID-19
12849	Czechia	CZE	Europe	10701777	cases	656	2020-28	14.63308383	13216	TESSy COVID-19
12850	Czechia	CZE	Europe	10701777	cases	771	2020-29	13.33423412	13987	TESSy COVID-19
12851	Czechia	CZE	Europe	10701777	cases	1375	2020-30	20.05274451	15362	TESSy COVID-19
12852	Czechia	CZE	Europe	10701777	cases	1454	2020-31	26.43486217	16816	TESSy COVID-19
12853	Czechia	CZE	Europe	10701777	cases	1567	2020-32	28.22895674	18383	TESSy COVID-19
12854	Czechia	CZE	Europe	10701777	cases	1657	2020-33	30.12583798	20040	TESSy COVID-19
12855	Czechia	CZE	Europe	10701777	cases	1907	2020-34	33.30288045	21947	TESSy COVID-19
12856	Czechia	CZE	Europe	10701777	cases	2449	2020-35	40.70352055	24396	TESSy COVID-19

Bibliography - References

[0]: ***Wikipedia***

[1]: *Mittelhammer, Ron C.; Miller, Douglas J.; Judge, George G. (2000). Econometric Foundations. Cambridge: Cambridge University Press. pp. 197–198. ISBN 0-521-62394-4.*

[2]: *Levenberg, Kenneth (1944). "A Method for the Solution of Certain Non-Linear Problems in Least Squares". Quarterly of Applied Mathematics. **2** (2): 164–168. doi:10.1090/qam/10666.*

[3]: *Marquardt, Donald (1963). "An Algorithm for Least-Squares Estimation of Nonlinear Parameters". SIAM Journal on Applied Mathematics. **11** (2): 431–441. doi:10.1137/0111030. hdl:10338.dmlcz/104299.*

[4]: *Girard, André (1958). "Excerpt from Revue d'optique théorique et instrumentale". Rev. Opt. **37**: 225–241, 397–424.*

[5]: *Wynne, C. G. (1959). "Lens Designing by Electronic Digital Computer: I". Proc. Phys. Soc. Lond. **73** (5): 777–787. Bibcode:1959PPS....73..777W. doi:10.1088/0370-1328/73/5/310.*

[6]: *Morrison, David D. (1960). "Methods for nonlinear least squares problems and convergence proofs". Proceedings of the Jet Propulsion Laboratory Seminar on Tracking Programs and Orbit Determination: 1–9.*

[7]: *Wiliamowski, Bogdan; Yu, Hao (June 2010). "Improved Computation for Levenberg–Marquardt Training" (PDF). IEEE Transactions on Neural Networks and Learning Systems. **21** (6).*

[8]: *Julier, S.J.; Uhlmann, J.K. (2004). "Unscented filtering and nonlinear estimation" (PDF). Proceedings of the IEEE. **92** (3): 401–422. doi:10.1109/jproc.2003.823141. S2CID 9614092.*

[9]: *Courses, E.; Surveys, T. (2006). Sigma-Point Filters: An Overview with Applications to Integrated Navigation and Vision Assisted Control. Nonlinear Statistical Signal Processing Workshop, 2006 IEEE. pp. 201–202. doi:10.1109/NSSPW.2006.4378854. ISBN 978-1-4244-0579-4. S2CID 18535558.*

[10]: IoT Business News, 2020, Global IoT Device Connections to Reach 11.7 billion in 2020, Surpassing Non-IoT Devices for the First time. Available online: https://iotbusinessnews.com/2020/11/20/03121-global-iot-device-connections-to-reach-11-7-billion-in-2020-surpassing-non-iot-devices-for-the-first-time/ (accessed on 20 November 2020).

[11]: International Data Corporation, 2020. IoT Growth Demands Rethink of Long-Term Storage Strategies. Available online: https://www.eetasia.com/iot-growth-demands-rethink-of-long-term-storage-strategies/ (accessed on 29 July 2020).

[12]: Cisco White Paper, 2015. Fog Computing and the Internet of Things: Extend the Cloud to Where the Things Are. Available online:

https://www.cisco.com/c/dam/en_us/solutions/trends/iot/docs/computing-overview.pdf (accessed on 20 November 2020).

[13]: Nguyen, B.M.; Binh, H.T.T.; Anh, T.T.; Son, D.B. Evolutionary Algorithms to Optimize Task Scheduling Problem for the IoT Based Bag-of-Tasks Application in Cloud–Fog Computing Environment. Appl. Sci. ***2019****, 9, 1730. [Google Scholar] [CrossRef] [Green Version]*

[14]: Pham, X.-Q.; Man, N.D.; Tri, N.D.T.; Thai, N.Q.; Huh, E.-N. A cost- and performance-effective approach for task scheduling based on collaboration between cloud and fog computing. Int. J. Distrib. Sens. Netw. ***2017****, 13. [Google Scholar] [CrossRef] [Green Version]*

[15]: Klonoff, D.C. Fog Computing and Edge Computing Architectures for Processing Data From Diabetes Devices Connected to the Medical Internet of Things. J. Diabetes Sci. Technol. ***2017****, 11, 647–652. [Google Scholar] [CrossRef] [PubMed] [Green Version]*

[16]: X.-S. Yang; S. Deb (December 2009). Cuckoo search via Lévy flights. World Congress on Nature & Biologically Inspired Computing (NaBIC 2009). IEEE Publications. pp. 210–214. arXiv:1003.1594v1.

[17]: Inderscience (27 May 2010). "Cuckoo designs spring". Alphagalileo.org. Retrieved 2010-05-27.

[18]: R. B. Payne, M. D. Sorenson, and K. Klitz, The Cuckoos, Oxford University Press, (2005).

[19]: Goldberg, David E. (1991). "The theory of virtual alphabets". Parallel Problem Solving from Nature. Parallel Problem Solving from Nature, Lecture Notes in Computer Science. Lecture Notes in Computer Science. Vol. 496. pp. 13–22. doi:10.1007/BFb0029726. ISBN 978-3-540-54148-6.

[20]: Janikow, C. Z.; Michalewicz, Z. (1991). "An Experimental Comparison of Binary and Floating Point Representations in Genetic Algorithms" (PDF). Proceedings of the Fourth International Conference on Genetic Algorithms: 31–36. Archived (PDF) from the original on 9 October 2022. Retrieved 2 July 2013.

[21]: Patrascu, M.; Stancu, A.F.; Pop, F. (2014). "HELGA: a heterogeneous encoding lifelike genetic algorithm for population evolution modeling and simulation". Soft Computing. ***18*** *(12): 2565–2576. doi:10.1007/s00500-014-1401-y. S2CID 29821873.*

[22]: Baluja, Shumeet; Caruana, Rich (1995). Removing the genetics from the standard genetic algorithm (PDF). ICML. Archived (PDF) from the original on 9 October 2022.

[23]: Srinivas, M.; Patnaik, L. (1994). "Adaptive probabilities of crossover and mutation in genetic algorithms" (PDF). IEEE Transactions on System, Man and Cybernetics. ***24*** *(4): 656–667. doi:10.1109/21.286385. Archived (PDF) from the original on 9 October 2022.*

[24]: Zhang, J.; Chung, H.; Lo, W. L. (2007). "Clustering-Based Adaptive Crossover and Mutation Probabilities for Genetic Algorithms". IEEE Transactions on Evolutionary Computation. ***11*** *(3): 326–335. doi:10.1109/TEVC.2006.880727. S2CID 2625150.*

[25]: See for instance Evolution-in-a-nutshell Archived 15 April 2016 at the Wayback Machine or example in travelling salesman problem, in particular the use of an edge recombination operator.

*[26]: Goldberg, D. E.; Korb, B.; Deb, K. (1989). "Messy Genetic Algorithms : Motivation Analysis, and First Results". Complex Systems. **5** (3): 493–530.*

[27]: Gene expression: The missing link in evolutionary computation

[28]: Gareth James; Daniela Witten; Trevor Hastie; Robert Tibshirani (2013). An Introduction to Statistical Learning. Springer. p. 204.

[29]: Brank, Janez; Mladenić, Dunja; Grobelnik, Marko; Liu, Huan; Mladenić, Dunja; Flach, Peter A.; Garriga, Gemma C.; Toivonen, Hannu; Toivonen, Hannu (2011), "Feature Selection", in Sammut, Claude; Webb, Geoffrey I. (eds.), Encyclopedia of Machine Learning, Boston, MA: Springer US, pp. 402–406, doi:10.1007/978-0-387-30164-8_306, ISBN 978-0-387-30768-8, retrieved 2021-07-13

*[30]: Kramer, Mark A. (1991). "Nonlinear principal component analysis using autoassociative neural networks". AIChE Journal. **37** (2): 233–243. doi:10.1002/aic.690370209. ISSN 1547-5905.*

*[31]: Kratsios, Anastasis; Hyndman, Cody (2021). "NEU: A Meta-Algorithm for Universal UAP-Invariant Feature Representation". Journal of Machine Learning Research. **22** (92): 1–51. ISSN 1533-7928.*

[32]: Persello, Claudio; Bruzzone, Lorenzo (July 2014). "Relevant and invariant feature selection of hyperspectral images for domain generalization". 2014 IEEE Geoscience and Remote Sensing Symposium. IEEE: 3562–3565. doi:10.1109/igarss.2014.6947252. ISBN 978-1-4799-5775-0. S2CID 8368258.

*[33]: Hinkle, Jacob; Muralidharan, Prasanna; Fletcher, P. Thomas; Joshi, Sarang (2012). Fitzgibbon, Andrew; Lazebnik, Svetlana; Perona, Pietro; Sato, Yoichi; Schmid, Cordelia (eds.). "Polynomial Regression on Riemannian Manifolds". Computer Vision – ECCV 2012. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer. **7574**: 1–14. arXiv:1201.2395. doi:10.1007/978-3-642-33712-3_1. ISBN 978-3-642-33712-3. S2CID 8849753.*

*[34]: Yarotsky, Dmitry (2021-04-30). "Universal Approximations of Invariant Maps by Neural Networks". Constructive Approximation. **55**: 407–474. arXiv:1804.10306. doi:10.1007/s00365-021-09546-1. ISSN 1432-0940. S2CID 13745401.*

*[35]: Hauberg, Søren; Lauze, François; Pedersen, Kim Steenstrup (2013-05-01). "Unscented Kalman Filtering on Riemannian Manifolds". Journal of Mathematical Imaging and Vision. **46** (1): 103–120. doi:10.1007/s10851-012-0372-9. ISSN 1573-7683. S2CID 8501814.*

*[36]: Kratsios, Anastasis; Hyndman, Cody (June 8, 2021). "NEU: A Meta-Algorithm for Universal UAP-Invariant Feature Representation". JMLR. **22**: 10312. Bibcode:2015NatSR...510312B. doi:10.1038/srep10312. PMC 4437376. PMID 25988841*

[37]: Anna Jordanous (10 April 2014). "What is Computational Creativity?". Retrieved 7 January 2019.

[38]: "Cryptanalysis/Signals Analysis". Nsa.gov. 2009-01-15. Retrieved 2013-04-15