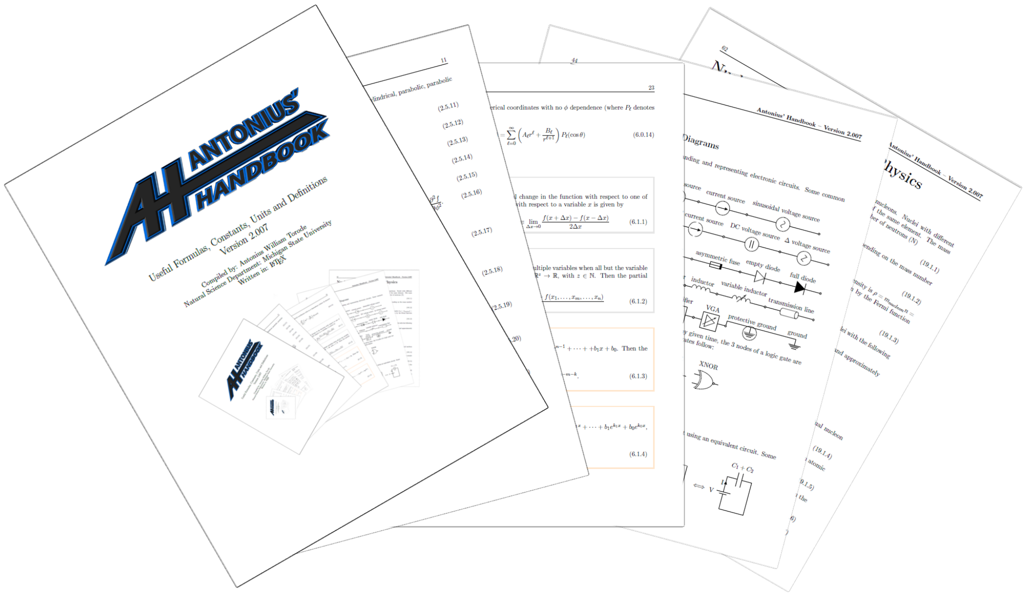
Antonius' Compendium

Compendium of Knowledge
Volume II: Neural Nucleus
Version 0.004

Compiled by: Antonius W. Torode[1]
1. Applied Research Laboratories - University Of Texas: Austin
Written in: LaTeX



Chief Editor: Antonius Torode

Published by Antonius Torode.

Hosted at: https://torodean.github.io/

Github Repository: https://github.com/torodean/Antonius-Compendium

Torode, A.
Antonius' Compendium.
Applied Research Laboratories –
June 19, 2025
Volume II.
Version: 0.004

Preface

This document is a compilation of ideas, scratch work, derivations, useful formulas, definitions, constants, and general information used for my own studies as a reference while furthering self education. These are my notes. It's purpose is to provide a complete 'compendium' per say of various ideas used often. All the material in this document was either directly copied from one of the references listed at the end or derived from scratch. On occasion *typos may exist* due to human error but will be corrected when discovered.

The version number is updated every time the document is distributed, printed for distribution, or a major update is added. This ensures that there is no two copies with different information and similar version numbers. The latest update date is automatically set to the current date each time the document is edited. Please refrain from distributing this handbook without permission from the original author/compiler. This book is formatted for printing.

Topics Covered In This Book

- Machine Learning
- Artificial Intelligence
- Modeling
- Psychology

The information in this book is in no way limited to the topics listed above. They serve as a simple guideline to what you will find within this document. For more information about this book or details about how to obtain your own copy please visit:

https://torodean.github.io/

"Scientific theories deal with concepts, not with reality. All theoretical results are derived from certain axioms by deductive logic. In physical sciences the theories are so formulated as to correspond in some useful sense to the real world, whatever that may mean. However, this correspondence is approximate, and the physical justification of all theoretical conclusions is based on some form of inductive reasoning." - Athanasios Papoulis (Probability, Random Variables, and Stochastic Processes book)

Disclaimer

This book contains formulas, definitions, and theorems that by nature are very precise. Due to this, some of the material in this book was taken directly from other sources such as but not limited to Wolfram Mathworld. This is only such in cases where a change in wording could cause ambiguities or loss of information quality. Following this, all sources used are listed in the references section and cited when used.

This page intentionally left blank.
(Yes, this is a contradiction.)

Contents

Modeling

1.1 Introduction

Modeling is the process of constructing abstract representations of real-world systems, behaviors, or phenomena. These representations—whether mathematical, statistical, or computational—enable analysis, simulation, prediction, and understanding of complex structures and dynamics.

This chapter will introduces several types of models used across disciplines, with a focus on those that capture structure, uncertainty, and sequential behavior. Topics include deterministic models, probabilistic frameworks, and data-driven models. The goal is to present models not only as tools for approximation or prediction, but as frameworks for organizing knowledge and reasoning about systems.

1.2 Markov Models

1.2.1 Overview

In a previous project I was working on for Dungeons & Dragons, I created a model for generating random names and character sequences. That project code can be found here:

`https://github.com/torodean/DnD/blob/main/templates/creator.py`.

The functionality was based on transition patterns between characters of preexisting names. I only later found out that this was referred to as a Markov model. The features related to this will be discussed here in a more generalized form. These models can be used to create elements which follow a similar pattern of an input sequence (such as creating predictive text).

1.2.2 First-Order Markov Model

This section contains an explanation of how the markov model functions. Consider some set of sequences S for some arbitrary type, where each element is denoted by a capitalized letter. For example, let

$$S = \{ABC, ABD, BAD\} \tag{1.2.1}$$

. The probability matrix P is constructed by determining all of the elements which follow another, and at what probability that element has of following the others (The probabilities of elements following some element K is P_K). that is $P(S) = \{K : P_K \forall K \in S\}$.

The set of elements which exist in S are $A, B, C, D, \emptyset$, where $\emptyset$ denotes the absence of an element (or beginning/end of a sequence). Starting with A, we can see that the A element is followed only by B (twice), and D (once) in S. The total number of elements ever following an A is thus three. The probabilities following an element A is thus

$$P_A = \begin{cases} B & : \text{twice} \\ D & : \text{once} \end{cases} \implies P_A = \begin{cases} B & : 66.\overline{6}\% \\ D & : 33.\overline{3}\% \end{cases} = \{B : 0.\overline{6}, D : 0.\overline{3}\} \tag{1.2.2}$$

Following this same process for the other elements gives

$$P_B = \{A : 0.\overline{3}, C : 0.\overline{3}, D : 0.\overline{3}\} \tag{1.2.3}$$

$$P_C = P_D = \{\emptyset : 1.0\} \tag{1.2.4}$$

$$P_\emptyset = \{A : 0.\overline{6}, B : 0.\overline{3}\}. \tag{1.2.5}$$

The total probability matrix for this set of sequences would then be

$$P(S) = \{A : P_A, B : P_B, C : P_C, D : P_D, \emptyset : P_\emptyset\} = \begin{cases} A : \{B : 0.\overline{6}, D : 0.\overline{3}\} \\ B : \{A : 0.\overline{3}, C : 0.\overline{3}, D : 0.\overline{3}\} \\ C : \{\emptyset : 1.0\} \\ D : \{\emptyset : 1.0\} \\ \emptyset : \{A : 0.\overline{6}, B : 0.\overline{3}\}. \end{cases} \tag{1.2.6}$$

The $\emptyset$ is a special case in that it represents the first character of a sequence (there is never a character after the last). This format may not look like a matrix at all, but it can be re-written to matrix format. First, note that there are a total of 5 elements (A, B, C, D, $\emptyset$) which will give a 5×5 matrix for all possible combinations. The matrix is configured such that both the rows and columns span from $A \to \emptyset$, covering all the elements of the set. The matrix value of a, b then represents the probability that element a will be proceeded by element b.

$$P(S) = \begin{bmatrix} 0 & 0.\overline{3} & 0 & 0 & 0.\overline{6} \\ 0.\overline{6} & 0 & 0 & 0 & 0.\overline{3} \\ 0 & 0.\overline{3} & 0 & 0 & 0 \\ 0.\overline{3} & 0.\overline{3} & 0 & 0 & 0 \\ 0 & 0 & 1.0 & 1.0 & 0 \end{bmatrix} \tag{1.2.7}$$

This probability matrix thus represents the probability of an element proceeding another in one of the given sequences. Each column of the matrix should total to 1.0, as they represent the total set of elements proceeding another. It can be used to generate new sequences which adhere to similar patterns of the input sequences. With larger data sets, more possibilities of sequences typically arise as probable outputs.

One important feature of these models is that under low-entropy (the model is derived from a deterministic source), a uniquely resolvable input set (You can reconstruct exactly one input set) and with enough metadata (initial state, model size, model order, etc), the model can be used to reconstruct the original data.

1.2.3 A Larger Example

Consider the following set of sequences:

$$S = \{ABCDE, ABDEE, AABCD, ACDCE, AABCD\} \tag{1.2.8}$$

The set of all elements of this sequence are $\{A, B, C, D, E, \emptyset\}$. There are a total of 6 elements. This set of sequences has 7 A's, 4 B's, 5 C's, 5 D's, 4 E's, and 5 $\emptyset$'s (beginning of each sequence). The probabilities P_k are then

$$P_A = \{A : 2/7, B : 4/7, C : 1/7, D : 0, E : 0, \emptyset : 0\} \tag{1.2.9}$$
$$P_B = \{A : 0, B : 0, C : 3/4, D : 1/4, E : 0, \emptyset : 0\} \tag{1.2.10}$$
$$P_C = \{A : 0, B : 0, C : 0, D : 4/5, E : 1/5, \emptyset : 0\} \tag{1.2.11}$$
$$P_D = \{A : 0, B : 0, C : 1/5, D : 0, E : 2/5, \emptyset : 2/5\} \tag{1.2.12}$$
$$P_E = \{A : 0, B : 0, C : 0, D : 0, E : 1/4, \emptyset : 3/4\} \tag{1.2.13}$$
$$P_\emptyset = \{A : 5/5, B : 0, C : 0, D : 0, E : 0, \emptyset : 0\} \tag{1.2.14}$$

This gives a probability matrix $P(S)$ of

$$P(S) = \begin{bmatrix} 2/7 & 0 & 0 & 0 & 0 & 5/5 \\ 4/7 & 0 & 0 & 0 & 0 & 0 \\ 1/7 & 3/4 & 0 & 1/5 & 0 & 0 \\ 0 & 1/4 & 4/5 & 0 & 0 & 0 \\ 0 & 0 & 1/5 & 2/5 & 1/4 & 0 \\ 0 & 0 & 0 & 2/5 & 3/4 & 0 \end{bmatrix} \tag{1.2.15}$$

1.2.4 Predicting Input Size

A probability matrix for a markov model is generated with a set of sequences. Each sequence s has a size, and the set of all sequences S also has a size. If the size of each sequence s is fixed, there will always be a minimum possible size of the S needed to generate the probability matrix. This analysis will being by using a fixed size for all $s \in S$.

Given an input probability matrix P, a limited prediction of the input data size can be made. Suppose $P(S)$ forms an $n \times n$ matrix, implying that it contains n elements. The matrix is formed by some number of combinations of these n elements. For visualization, suppose the matrix looks something like the following:

$$P(S) = \begin{bmatrix} s_{00} & s_{10} & s_{20} & \cdots & s_{n0} \\ s_{01} & s_{11} & s_{21} & \cdots & s_{n1} \\ s_{02} & s_{12} & s_{22} & \cdots & s_{n2} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ s_{0n} & s_{1n} & s_{2n} & \cdots & s_{nn} \end{bmatrix} \tag{1.2.16}$$

As mentioned before, the probabilities of each column k (each column represents a different element of S, so k here, is just any element $k \in S$) must add to 1, giving

$$P(k) = \sum_{i=0}^{n} s_{in} = 1, \forall k \in S. \tag{1.2.17}$$

Every element $k \in S$ will appear an integer number of times (since an element either appears or does not). This means, that every probability s will be a rational value, $s \in \mathbb{Q}$ (since it is defined as the total occurrences of k over the total possible occurrences of any element). Therefore, we can define $s^*_{ik} \in \mathbb{N}$ such that $s^*_{ik} = s_{ik}N_k$, where $N_k \in \mathbb{N}$. Given that N_k is the least common denominator (LCD) when summing all values of s_{ik} to 1, this value of N_k would then be the minimum possible number of total elements of k, denoted T_k, in the initial sequence which produced this matrix. This gives

$$\min(T_k) = \sum_{i=0}^{n} s^*_{ik} = N_k, \forall k \in S. \tag{1.2.18}$$

Note that the $\emptyset$ element is a special case here. It is defined as the start or end of a sequence and therefore there will always be n of them from the reference of following an element, and n from the reference of preceding an element.

This is more of an observational conclusion than a formal theorem. Notice in equation 1.2.7, for the first column, we have $\frac{2}{3}$ and $\frac{1}{3}$ as the probabilities. When summed to 1, the LCD of these is 3. This correctly matches the value of the number of A elements (which corresponds to the first column) in the corresponding sequence of S which was used. However, if the initial sequence had 6 elements, and the probabilities were thus $\frac{4}{6} = \frac{2}{3}$ and $\frac{2}{6} = \frac{1}{3}$ (respectively), then the value would have been incorrect because the fractions may have been reduced. This is the reason I say it is $\min(T_k)$ rather than T_k. Given that the fractions for the probabilities are left in an non-reduced form, or are irreducible to begin with, this would be T_k. But for general purposes, the probabilities are likely always reduced to simplest form and thus we can only determine a minimum value of T_k. It will be true that the total number is some multiple of T_k, however, which can be valuable information.

There may be some interesting insights to gain from the rows of the probability matrix as well. Given the existence of N_k, which has already been established, the sum of the row R for some element k can be written as

$$R(k) = \sum_{i=0}^{n} s_{ki} = \sum_{i=0}^{n} \frac{s^*_{ki}}{N_i} = \frac{s^*_{k0}}{N_0} + \frac{s^*_{k1}}{N_1}\frac{s^*_{k2}}{N_2} + \cdots + \frac{s^*_{kn}}{N_n}. \tag{1.2.19}$$

If N_i is an unreduced LCD for the values of its row, then these coefficients give the total number of elements k in the set of sequences

$$T_k = \sum_{i=0}^{n} s^*_{ki}. \tag{1.2.20}$$

Again, this is more of an observational conclusion than a formal theorem. But this can be seen in the following way. Each element, when appearing in a sequence, will add some minimum amount of probability p_{min} to a corresponding position in the matrix. this amount will always be equal to $1/T_k$. If the elements appear after another element more than once, say, s^*_{ki} times, then this coefficient increments, giving s^*_{ki}/T_k. For unreduced fractional forms of each probability, $T_k = N_k$, so s^*_{ki}/N_k is just a count of how many elements there are. Thus, by summing these counters for each case (an element proceeding another element), we have the total number of elements in a set of sequences.

! Section in Progress !

1.2.5 Generating A New Sequence

One primary use for generating Markov models is so that they can be used to generate a sequence adhering to a similar pattern of the system that the model was generated from. This is useful in many application. One trivial example, which I've used this for in the past, is generating new fantasy-style (Dwarven, Elvish, etc) names based on lists of pre-existing names. In this example case, this type of model served as the perfect tool for generating new and unique names. The newly generated names followed similar stylistic and phonetic patterns of the names from the original list, which gave for a very realistic set of unique names.

When generating a new sequence based on an existing model, a few considerations are needed. The size of the generated sequence needs considered. This can either be fixed (for instance in the case of a set of fixed-sized sequences), randomized (based on some criteria). Typically, I have found that creating a randomized value between the minimum sequence size and maximum sequence size is usually most representative of the data (with some normalization factors for more confined results).

The procedure for calculating subsequent sequence elements can also be done in a few ways. First, the simplest method is to simply use the exact probabilities from the model, and generate a random subsequent element based on the model weights. There are a few cases which could arise where this type of generation would yield undesirable sequences. First, if you rely solely on the generated probabilities, you will never be able to get results outside of those. For example, for the sequence found in 1.2.1, you will never get an A followed by a C because it does not ever exist. Alternately, you may have a rather large dataset with a few outliers. For example, a large list of words may contain something like "syzygy". The sequences generated from this could then contain words with "yzy", which may not be desirable for any other sequences but this one. In cases like this, some of the smaller probabilities can be ignored (which usually works nicely for larger data sets).

With the above optional conditions in mind, the steps for generating a sequence follow:

1. Determine the first element of the sequence.

2. Use the probability matrix of the first element, with appropriate conditional changes, to generate the second element. This probability is based on the elements following the previous element.

3. Repeat the previous step until the desired sequence size is reached...

The transition from one element E_1 to another element E_2 is denoted via $E_1 \rightarrow E_2$. Consider the sequences given in 1.2.8, whose probability matrix is given in 1.2.15. In this example, to begin a sequence is to begin with the element $\emptyset$ (representing the start of a sequence). The only element which ever follows this one is A with 100% probability. So sticking to the given probabilities, a new sequence would begin as $\emptyset \rightarrow A$. Continuing this pattern, the elements which can follow A are A, B, or C. Suppose we randomly choose C, which gives the new sequence $\emptyset \rightarrow A \rightarrow C$. Continuing this patter could lead to a final sequence (as an example) of $ACEED$. Thus, a new, perfectly valid sequence is generated using the probability matrix.

Repeating this pattern can give any number of elements, though some issues may arise which require some edge-case corrections to complete the sequence. For example, suppose we have a sequence starting with $ABCD$. Since $D \rightarrow C$ is perfectly valid, this sequence could continue as $ABCDC$. However, this sequence is already five elements long, and note that $C \rightarrow \emptyset$ is not a valid transition. Therefore, if we strictly held to the rules of this probability matrix, $ABCDC$ would be troublesome in that it is either non-valid or would have to break the rules allowing $C \rightarrow \emptyset$. With these models, certain rules like this often need *broken* in order to provide a path to completion for some sequences (though this is entirely a preferential choice depending on the use).

1.2.6 Higher-Order Markov Model

A higher order Markov model follows similar principals as a first-order Markov model. The difference is how many elements in sequence are tracked and considered when determining another. Consider the same sequence found in 1.2.1. Suppose the order of this model is O (that is a O-order Markov model). This would mean that instead of calculating the probability that an element follows another, the probability matrix would instead contain the probability of an element following any sequence of elements from 1 to O. For example, suppose we want $O = 2$, giving a second order Markov model.

There are actually multiple ways to construct the probability matrix depending on the intended purpose of the model. For the sequence in 1.2.1, one possible matrix can be constructed from the possible combinations of any sequence of 1 *or* 2 elements, which are the same elements in the first-order Markov model with the union of the additional probabilities of every sequence of 2 elements. The sequence of two elements within S in this case are $\emptyset A$ (twice), $\emptyset B$ (once), AB (twice), BC (once), BD (once), BA (once), AD (once), $C\emptyset$ (once) and $D\emptyset$ (twice). The probability for a second order is denoted by P^2. This can be written as

$$P_{\emptyset A} = \{A : 0, B : 2/2, C : 0, D : 0, \emptyset : 0\} \tag{1.2.21}$$
$$P_{\emptyset B} = \{A : 1/1, B : 0, C : 0, D : 0, \emptyset : 0\} \tag{1.2.22}$$
$$P_{AB} = \{A : 0, B : 0, C : 1/2, D : 1/2, \emptyset : 0\} \tag{1.2.23}$$
$$P_{BA} = \{A : 0, B : 0, C : 0, D : 1/1, \emptyset : 0\} \tag{1.2.24}$$
$$P_{AD} = \{A : 0, B : 0, C : 0, D : 0, \emptyset : 1/1\} \tag{1.2.25}$$
$$P_{BC} = \{A : 0, B : 0, C : 0, D : 0, \emptyset : 1/1\} \tag{1.2.26}$$
$$P_{BD} = \{A : 0, B : 0, C : 0, D : 0, \emptyset : 1/1\} \tag{1.2.27}$$
$$\tag{1.2.28}$$

The complete matrix P^2 could be constructed by taking this and the union of the matrix given in 1.2.7. It could also be constructed into its own matrix without any consideration for single element transitions. A third approach is to construct this matrix with the union of only the single element transitions which don't overlap with the two-element sequences. For example, the elements of $\emptyset$ (which denotes the start of a sequence), would not be a two-element sequence, but rather just a single element, which is then followed by either A or B. This would be $P_{\emptyset}$, which could be included in the above list. In this example, there are no other cases, but other examples would have more.

Taking the union of all individual element sequences with the multi-element sequences could be a preferred way of constructing these probability matrices for a more complete model because they account for all the multi-element sequences, and then also provide individual element sequences if those do not exist. This is particularly useful when generating new sequences. If you are generating a new sequence, it could be common that you come across a sequence of elements which do not have an existing element proceeding it. In this case, you could shorten the element by one, and check against that sequence to determine the next.

Artificial Intelligence

2.1 Introduction

Artificial Intelligence (AI) is the field of study concerned with the design and development of systems capable of performing tasks that typically require human intelligence. These tasks include learning, reasoning, problem-solving, perception, and language understanding. In a sense, this is an attempt to artificially mimic human intelligence while simultaneously combining it with the advantages that modern technological computing powers bring. AI spans a wide range of sub-fields, from symbolic logic and knowledge representation to machine learning and neural networks. It intersects with disciplines such as computer science, mathematics, neuroscience, and philosophy.

Modern AI systems are broadly categorized into two classes: *narrow AI*, designed for specific tasks, and *general AI*, which aspires to emulate human-level cognition across diverse domains. Key concepts include algorithms, data structures, optimization, statistical inference, and computational models of learning. Recent advancements in Large Language Models (LLMs) have demonstrated the ability of transformer-based architectures to generate coherent text, perform reasoning tasks, and interface with complex domains using natural language, which gives an appearance for the foundations of creating a *general AI*. Artificial General Intelligence (AGI) refers to a type of AI that possesses the ability to understand, learn, and apply knowledge across a wide range of tasks at a level comparable to human intelligence, which is the goal of many.

Applications of AI are pervasive, influencing medicine, finance, robotics, language processing, and decision-making systems. Continued advancement in AI raises important technical, ethical, and societal questions, many of which remain open areas of research.

Encryption

3.1 Introduction

Encryption is a fundamental technique in information security that transforms readable data, usually known as plain text, into an unreadable format, called cipher text, using a cryptographic algorithm and a key. The primary goal of encryption is to ensure confidentiality, preventing unauthorized access to sensitive information. There are two main types of encryption: symmetric-key encryption, where the same key is used for both encryption and decryption, and asymmetric-key encryption, which uses a pair of public and private keys. Encryption plays a crucial role in securing communication, data storage, and digital transactions.

3.2 d0s3 Encryption

3.2.1 Introduction and Purpose

The d0s3 encryption algorithm is a continuation of an earlier personal encryption project originally created as part of the D3C (d0sag3[1] command) program many years ago. The previous versions, d0s1 and d0s2, were developed during initial experimentation with C++ and served as educational implementations. This document focuses solely on the redevelopment of d0s3. Unlike its predecessors, d0s3 is intended to implement a new, more advanced encryption method capable of handling full file encryption or larger data sets in general. This section contains brainstorming, design notes, and implementation details as development resumes on the d0s3 algorithm.

The purpose of the d0s3 algorithm is to securely encrypt data in such a way that it requires an asymmetric encryption key to decrypt the data (such as a password). There are a few initial ideas and concepts which will be explored in developing this encryption algorithm, which will be explained. The algorithm must be such that any size of data can easily be encrypted in both a secure and efficient way.

3.2.2 Historical Design

The original version of the d0s3 encryption was based on storing the bits of information within a Rubik's cube style data structure, and scrambling the cube in order to encrypt the data. This was a rather simple and straight forward approach with a few major flaws. The biggest flaw here is that the data structure of the Rubik's cube increases in size by a fixed N^3, where N is the number of bits to encrypt. This is not always conducive to being efficient with the data size or storage. This new design will further explore a similar method of storing the data in a cube-like structure, but explore storing the bits in differing manners and in higher dimensions (which could allow variable storage types to adjust to different sizes). Section 3.2.3 will contain this exploration.

3.2.3 Exploring the Cubes

Consider the construction of a sphere in multiple dimensions (shown in figure 3.1). In 0-dimension, there is simply a point. Increasing the dimension by 1 causes the number of points to double. In 1-dimension, this gives 2 points. The number of points doubles with each new dimension, giving 2^d points (where $d \in \mathbb{N}$ is

[1]The alias d0sag3 comes from my xbox live gamer-tag from when I was younger. I used this alias to create many projects when I was first beginning to learn programming, graphic design, and other things.

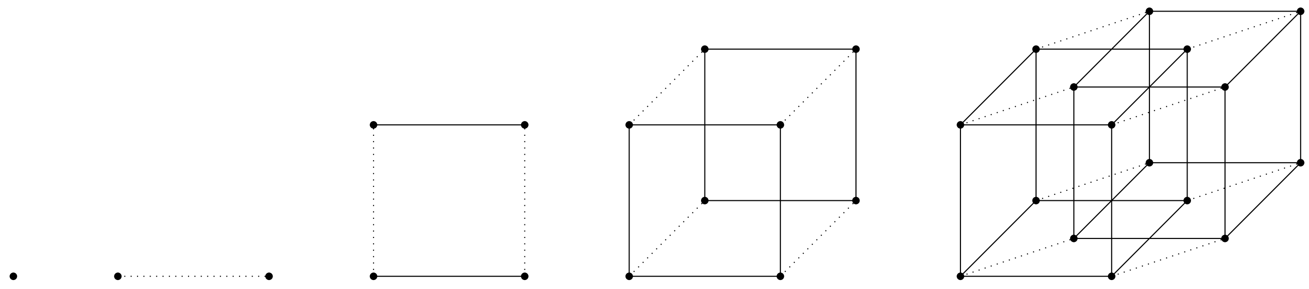


Figure 3.1: This figure shows the transition from a single 0-dimensional point to a 4-dimensional tesseract. At each transition, the previous is duplicated (shown in bold lines and points), and the connections from the previous dimension to the new are displayed with dotted lines.

the dimension scale). This gives the sequence of points per dimension as 1,2,4,8,16,... When the points in 1-dimension are connected, this makes 1 line segment. The pattern continues as the dimensions increment.

$$\text{points}(d) = \mathcal{P}_d = 2^d \tag{3.2.1}$$

In 2-dimension, the 2 points from the 1-dimension are duplicated, giving 4 points. The line is also duplicated (in a sense), and new line segments are created to connect the newly created points to the old points. This pattern of line segments also repeats, where the number of new line segments will always be the number of points that existed in the previous dimension (since each point will connect to another new point). Then, the total line segments will be this value summed with the number of line segments from the previous dimension times two (since they are duplicated). Therefore, the number of lines ℓ in a dimension is $\mathcal{L}_d = 2\mathcal{L}_{d-1} + 2^{d-1}$. This gives the sequence of 0,1,4,12,32,... When these line segments are connected in 2-dimension, they form a square.

$$\text{lines}(d) = \mathcal{L}_d = 2\mathcal{L}_{d-1} + \mathcal{P}_{d-1} = 2\mathcal{L}_{d-1} + 2^{d-1} \tag{3.2.2}$$

In 2-dimensions and higher, the geometric element of a square emerges. The square is formed by a combination of 4 lines forming the edges and 4 points forming the corners. In each higher dimension, the squares from the previous dimension are doubled, and then the number of squares increased by the number of lines from the previous dimension. The pattern gives 0,1,6,24,80... When these squares are connected, they form the 3-dimensional cube. The number of squares follows as

$$\text{squares}(d) = \mathcal{S}_d = 2\mathcal{S}_{d-1} + \mathcal{L}_{d-1} = 2\mathcal{S}_{d-1} + 2\mathcal{L}_{d-2} + 2^{d-2} \tag{3.2.3}$$

This pattern continues for each new geometric element which emerges from the higher dimensions. Therefore, the number of cubes is

$$\text{cubes}(d) = \mathcal{C}_d = 2\mathcal{C}_{d-1} + \mathcal{S}_{d-1} = 2\mathcal{C}_{d-1} + 2\mathcal{S}_{d-2} + 2\mathcal{L}_{d-3} + 2^{d-3} \tag{3.2.4}$$

All of these values for each geometric element are determined based on the values which occur in the dimensions prior. Since the first geometric element (the point) only depends on the dimension, it should be possible to represent the others in terms of only the dimension as well. Writing out the sequences, we can begin to see the patterns outlined in the above sequences emerge:

dimension	0D	1D	2D	3D	4D	5D
points	1	2	4	8	16	32
lines	0	1	4	12	32	80
squares	0	0	1	6	24	80
cubes	0	0	0	1	8	40
tesseracts	0	0	0	0	1	10

(3.2.5)

A few observations from the above sequences and corresponding formulas are important. First, the row of 1's at the start of the sequence in a diagonal pattern, as well as the continual addition of terms from the previous sequence in each other sequence brings to mind the Pascal Triangle. Recall Pascal's triangle, where each element is determined by the sum of the 2 nearest elements above it. These values give the binomial coefficients of two values a and b as $\binom{a}{b}$.

$$\begin{array}{ccccccccccc}
 & & & & & 1 & & & & & \\
 & & & & 1 & & 1 & & & & \\
 & & & 1 & & 2 & & 1 & & & \\
 & & 1 & & 3 & & 3 & & 1 & & \\
 & 1 & & 4 & & 6 & & 4 & & 1 & \\
1 & & 5 & & 10 & & 10 & & 5 & & 1
\end{array} \tag{3.2.6}$$

By writing the pattern such that the values of each is represented as a factored value below it gives the following pattern:

dimension	0D	1D	2D	3D	4D	5D
points	1	2	4	8	16	32
	(1)	(1)2	(1)4	(1)8	(1)16	(1)32
lines		1	4	12	32	80
		(1)	(2)2	(3)4	(4)8	(5)16
squares			1	6	24	80
			(1)	(3)2	(6)4	(10)8
cubes				1	8	40
				(1)	(4)2	(10)4
tesseracts					1	10
					(1)	(5)2

(3.2.7)

From this, it's easy to see that the values from Pascal's Triangle emerge as a factor for each value. These values are based on the dimensions d and row i. The coefficients matching pascals triangle are then

$\binom{d}{i}$. The remaining factor is simply the number of points in each dimension offset by i, giving a factor of 2^{d-i}. Since the rows represent the various geometric sequences, we could assign them to some incremental element starting 0 (representing points), and incrementing. Denote these values of the above table as $\mathbb{G}(d, i)$, where d is the dimension and i represents the geometric element being represented (e.g., $i = 0$ would be points, $i = 1$ would be lines, $i = 2$ would be squares, etc). The values then can be represented as

$$\mathbb{G}(d, i) = \binom{d}{i} 2^{d-i} \tag{3.2.8}$$

This gives the equations and values represented in the following table.

	0D	1D	2D	3D	4D	5D	...	dD
Points	1	2	4	8	16	32		2^d
Lines	0	1	4	12	32	80		$d \cdot 2^{d-1}$
Squares	0	0	1	6	24	80		$\binom{d}{2} \cdot 2^{d-2}$
Cubes	0	0	0	1	8	40		$\binom{d}{3} \cdot 2^{d-3}$
Tesseracts	0	0	0	0	1	10		$\binom{d}{4} \cdot 2^{d-4}$

Table 3.1: Number of geometric elements in the various dimensions d by constructing cube shapes in higher and lower dimensions.

Psychology

4.1 Introduction

Psychology is the scientific study of mind and behavior, encompassing the processes underlying cognition, emotion, perception, and action. It explores how individuals think, feel, and behave across various contexts, integrating insights from biology, sociology, philosophy, and neuroscience. The field is broadly divided into sub-disciplines such as cognitive psychology, behavioral psychology, developmental psychology, clinical psychology, and social psychology. Each examines different aspects of mental processes and behavior, using both experimental and observational methods.

Psychological research informs fields like education, mental health, artificial intelligence, and human-computer interaction, providing foundational understanding of human cognition and behavior.