Winter 2015

THE APPLICATION OF THE EXPECTATION MAXIMIZATION ALGORITHM ONTO BIG DATA

Jason Beffel
John Carroll University, jasonbeffel@gmail.com

THE APPLICATION OF THE EXPECTATION MAXIMIZATION ALGORITHM ONTO BIG DATA

An Essay Submitted to the
Office of Graduate Studies
College of Arts & Sciences of
John Carroll University
in Partial Fulfillment of the Requirements
for the Degree of
Master of Science in Mathematics

By
Jason M. Beffel
2015

1. Introduction

With the growing prevalence of big data, it is interesting to see whether the popular machine-learning tool known as the Expectation Maximization (EM) algorithm (Dempster, Laird, & Rubin, 1977) is applicable and beneficial for data analysis. Can the EM algorithm be used directly on big data without requiring additional changes or adjustments? Yes, the EM algorithm can be applied to a big data set modified or unmodified, and the goal of this paper is to show the technique used to do so.

The connection between big data and the EM algorithm can be seen through the lens of machine learning. Machine learning uses several different statistics algorithms that can "learn" from being applied to data sets (Kohavi & Provost 1998). This "learning" usually refers to inductive learning or pattern recognition/repetition, which is applied by the algorithms. Machine learning is a subfield of computer science and combines statistical algorithms and inductive learning to create artificial intelligence. Furthermore, machine learning is very closely related to data mining. Both methodologies rely heavily on the same foundation of statistics algorithms, but each has a different goal. While machine learning makes a prediction of an unknown variable based on known variable(s), data mining uncovers properties of the data based on the known variable(s) (Fayyad, Piatetsky-Shapiro, & Smyth, 1996). Coincidentally, these related fields are both making very good use of the increased access to big data.

The EM algorithm is one of the algorithms used frequently in machine learning. In the past, machine learning has only been used to analyze relatively small data sets. With the entrance of big data, statisticians have been able to unleash these machine learning algorithms on big data and discover things that were not possible before the advent of very large, accessible data sets (Olsen, 2013). For example, applying machine learning to big data is a great step forward for business analytics (Rogers, 2015). However, there is still a significant human component required for these processes. For instance, data sets and the rules regulating each algorithm are so complex that the analyst must determine which machine learning algorithm can be used or, if any can be used on big data at all. Speculation regarding whether these algorithms can be used on big data has led statisticians to question how many of the machine learning algorithms can be easily applied to big data. To carry out a more detailed assessment of how machine learning algorithms can be applied to the analysis of big data, the remainder of this document will focus specifically on what is known as the EM algorithm.

The main concern regarding the direct application of the EM algorithm on big data is that the EM algorithm was explored with incomplete data (Dempster et al., 1977). Consequently, while the E-step can be finished with ease, the M-step is much more complicated to complete with big data because of the immense size and completeness available from big data (Wolfe, Haghighi, & Klein, 2008).

In order to investigate the full relationship between the EM Algorithm and big data, first one must explore both of these topics individually to understand the basis of where each might possibly connect to the other. The following sections will provide an overview of each topic, including information as to how they could potentially relate. The goal of this analysis is to determine if the EM Algorithm can be appropriately applied to big data, and if so, how this application is possible. The research, thus far, indicates that in order to be able to use the EM algorithm on big data, statisticians are combining Topology with Data Analysis to make this possible (Wolfe et al., 2008).

2. Expectation Maximization Algorithm

The Expectation Maximization (EM) algorithm is a tool used to compute a maximum likelihood estimation of incomplete data, with unknown or latent variables. There are a few variations of this algorithm as it is used for a number of distinct applications. The EM algorithm is a popular tool in machine learning and alternately is used by statisticians to solve problems where certain parameters of data must be estimated by assuming that unknown data is known (Borman, 2004; Dempster et al., 1977). The working of this algorithm can be broken down into two steps or processes: the expectation (E-step) and the maximization (M-step). Obviously, these steps are how the EM Algorithm derives its name.

The EM Algorithm was first named and described in depth in a paper by Dempster, Laird and Rubin (Dempster et al., 1977). The premise of the algorithm is to maximize the likelihood of an estimation of unknown variables or data through several cycles of the algorithm. To do this, the natural log of the likelihood function of a given probability distribution must be maximized.

Before moving on to discuss the application and derivation of the EM Algorithm, it is critical to first review some important concepts and definitions which will aid in the understanding of the EM Algorithm.

Definition: Likelihood function

Given a parameter ϕ and a data set $\overline{X}$, the likelihood function $L(\phi)$ is equal to the probability $P(\overline{X} \mid \phi)$ of that data set. This is not to be confused with conditional probability even though the notations are similar (Harris & Stocker, 1998; Weisstein, 2015).

In the case that the data set is a finite sample with a known density function, the likelihood can also be written as a product of the probability of each point in the sample. If the fixed data set $\overline{X} = \{x_1, x_2, ..., x_n\}$ for some $n \in N$, then the likelihood function is: $L(\phi) = f(x_1 \mid \phi) f(x_2 \mid \phi) ... f(x_n \mid \phi) = \prod_{i=1}^{n} f(x_i \mid \phi)$.

Definition: Log Likelihood function

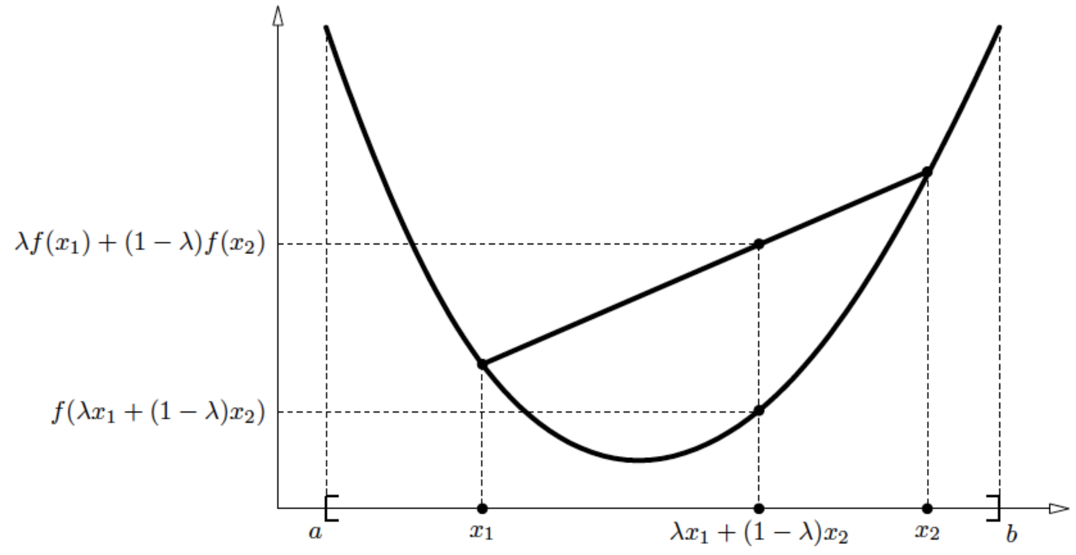
The natural log of the Likelihood function of a data set $\overline{X}$ and a given parameter ϕ is represented by: $l(\phi) = \ln(L(\phi))$.

The initial derivation of the EM Algorithm relies on the key facts that define the natural log of the likelihood function. This composition of functions is convex, meaning it allows for the use of Jensen's Inequality.

Definition: Convex function

Let f be a real valued function defined on an interval $I = [a,b]$. f is said to be convex on I if

$\forall x_1, x_2 \in I, \quad \lambda \in [0,1] \Rightarrow f(\lambda x_1 + (1-\lambda) x_2) \leq \lambda f(x_1) + (1-\lambda) f(x_2)$.



$\lambda f(x_1) + (1-\lambda) f(x_2)$
$f(\lambda x_1 + (1-\lambda)x_2)$
a
x_1
$\lambda x_1 + (1-\lambda)x_2$
x_2
b

Figure 2.1: An example of a convex function (Borman, 2004).

Definition: Concave function

A function f is concave if $-f$ is convex.

Theorem: Suppose $f''(x)$ exists on $x \in I = (a,b)$. If $f''(x) \geq 0$ on $\forall x \in I$ then f is convex (Borman, 2004).

Note: $\text{If } x \in (0,\infty), \text{ then } f(x) = -\ln(x)$ is convex by the above theorem.

Theorem: (Jensen's Inequality) Let μ be a measure on X and $f : X \to IR.$ If $\mu(X) = 1, f$ is Lebesgue integrable, $f(X) \subseteq I = (a,b)$, $\phi : I \to IR$ and is convex, then

$$\phi\left(\int_X f\, d\mu\right) \leq \int_X (\phi \circ f)\, d\mu$$

Jensen's Inequality has several different possible applications to functions. The following is a list of some results:

2.1) If f is Lebesgue integrable over $[0,1]$, then $\exp\left(\int_0^1 f(x)dx\right) \le \int_0^1 \exp(f(x))dx$.

2.2) $\forall n \in \mathrm{N}, \forall x_1, x_2, ..., x_n \in IR, \forall \alpha_1, \alpha_2, ..., \alpha_n \in [0,\infty), \sum_{i=1}^{n} \alpha_i = 1 \Rightarrow \exp\left(\sum_{i=1}^{n} \alpha_i x_i\right) \le \sum_{i=1}^{n} \alpha_i e^{x_i}$.

2.3) $\forall n \in \mathrm{N}, \forall x_1, x_2, ..., x_n \in IR, \quad \exp\left(\frac{1}{n}\sum_{i=1}^{n} x_i\right) \le \frac{1}{n}\sum_{i=1}^{n} e^{x_i}$.

2.4) $\forall n \in \mathrm{N}, \forall y_1, y_2, ..., y_n \in [0,\infty), \forall \alpha_1, \alpha_2, ..., \alpha_n \in [0,\infty), \sum_{i=1}^{n} \alpha_i = 1 \Rightarrow y_1^{\alpha_1} y_2^{\alpha_2} ... y_n^{\alpha_n} \le \sum_{i=1}^{n} \alpha_i y_i$.

The following is a proof of a slightly more generalized version of Result 2.2 as it has an almost a direct application to the proof of the EM Algorithm:

Let $f : I \rightarrow IR$ and be convex with $I = [a,b]$.

If $\forall n \in \mathrm{N}, \forall x_1, x_2, ..., x_n \in I,$ and $\forall \alpha_1, \alpha_2, ..., \alpha_n \in [0,\infty),$ where $\sum_{i=1}^{n} \alpha_i = 1,$

Then $f\left(\sum_{i=1}^{n} \alpha_i x_i\right) \le \sum_{i=1}^{n} \alpha_i f(x_i)$

Proof:

For this proof, induction is used. First we must show that $f\left(\sum_{i=1}^{2} \alpha_i x_i\right) \le \sum_{i=1}^{2} \alpha_i f(x_i)$

We will use the method of induction where the base case is with n = 2 because the case where n = 1 is trivial. The use of the definition of convex functions in the proof below is vital when moving across the inequality, while the rest of the proof easily follows.

$$\begin{aligned}
f\left(\sum_{i=1}^{2}\alpha_i x_i\right) &= f(\alpha_1 x_1) + f(\alpha_2 x_2) \\
&= f(\alpha_1 x_1) + f((1-\alpha_1)x_2) \quad \text{since } \alpha_1 + \alpha_2 = 1 \text{ by assumption} \\
&\le \alpha_1 f(x_1) + (1-\alpha_1) f(x_2) \quad \text{by the definition of convex functions} \\
&= \alpha_1 f(x_1) + \alpha_2 f(x_2) \\
&= \sum_{i=1}^{2}\alpha_i f(x_i)
\end{aligned}$$

Now, assume that $f\left(\sum_{i=1}^{n}\alpha_i x_i\right) \le \sum_{i=1}^{n}\alpha_i f(x_i)$ is true. The goal is to prove that this is also true for the case for n+1. The proof of this case will flow similarly to the base case above, by utilizing the definition of convex functions as the key to the success of the proof (Borman, 2004).

$$\begin{aligned}
f\left(\sum_{i=1}^{n+1}\alpha_i x_i\right) &= f\left((\alpha_{n+1}x_{n+1}) + \sum_{i=1}^{n}\alpha_i x_i\right) \\
&= f\left((\alpha_{n+1})x_{n+1} + (1-\alpha_{n+1})\frac{1}{(1-\alpha_{n+1})}\sum_{i=1}^{n}\alpha_i x_i\right) \\
&\le (\alpha_{n+1}) f(x_{n+1}) + (1-\alpha_{n+1}) f\left(\frac{1}{(1-\alpha_{n+1})}\sum_{i=1}^{n}\alpha_i x_i\right) \\
&= (\alpha_{n+1}) f(x_{n+1}) + (1-\alpha_{n+1}) f\left(\sum_{i=1}^{n}\frac{\alpha_i}{(1-\alpha_{n+1})}x_i\right) \\
&\le (\alpha_{n+1}) f(x_{n+1}) + (1-\alpha_{n+1}) \sum_{i=1}^{n}\frac{\alpha_i}{(1-\alpha_{n+1})} f(x_i) \\
&= (\alpha_{n+1}) f(x_{n+1}) + \sum_{i=1}^{n}\alpha_i f(x_i) \\
&= \sum_{i=1}^{n+1}\alpha_i f(x_i)
\end{aligned}$$

Since $f(x) = \ln(x)$ is concave when $x \in [0, \infty)$, Jensen's inequality can be applied to this function to obtain a property: $\ln\left(\sum_{i=1}^{n} \alpha_i x_i\right) \geq \sum_{i=1}^{n} \alpha_i \ln(x_i)$. This property will be further utilized at a later step.

The EM Algorithm deals with samples where the parameter ϕ is not known. Since the natural log function is an increasing function, the value of ϕ that maximizes the likelihood function will be the same value that is maximized in the Log Likelihood function. This characteristic makes using the Log Likelihood functions easier to work with.

2.1. Convergence of the EM Algorithm

The ultimate goal of the EM Algorithm is to find ϕ that maximizes the log likelihood function $l(\phi) = \ln(L(\phi))$. In the EM Algorithm, since the parameter ϕ of $l(\phi)$ is unknown, the E-step is where an estimation of ϕ is made. The E-step is followed by the M-step, which results in maximization after many iterations of the algorithm. After n iterations of the algorithm, ϕ_n represents the expected value of the log likelihood function from the n iteration to be used in the $n+1$ iteration, so that $l(\phi_{n+1})$ can be maximized, and solved for ϕ_{n+1} to complete the $n+1$ iteration of the EM Algorithm. This provides a value so that $l(\phi_{n+1}) > l(\phi_n)$, and

$$\begin{aligned} l(\phi) - l(\phi_n) &= \ln(L(\phi)) - \ln(L(\phi_n)) \\ &= \ln(P(\overline{X} \mid \phi)) - \ln(P(\overline{X} \mid \phi_n)) \end{aligned}$$

In the M-step, finding ϕ_{n+1} that completes the maximization of $l(\phi_{n+1})$ is the same as solving for the ϕ that maximizes the difference of $l(\phi) - l(\phi_n) = \ln\left(P(\overline{X} \mid \phi)\right) - \ln\left(P(\overline{X} \mid \phi_n)\right)$.

In the EM Algorithm, we are assuming that $\overline{X} = \{x_1, x_2, ..., x_m\}$ for some $m \in N$ is a fixed set. But up to this point, we have not considered missing variables. Since the EM Algorithm is used specifically to find maximum likelihood when there are missing variables, we will introduce a set that represents the set of unobserved values that complement our fixed set, $\overline{X}$. We will call this set $\overline{Y} = \{y_1, y_2, ...\}$. The set does not have to be discrete, though it can be. This allows the total probability of $\overline{X}$ to be written in terms of y as $P(\overline{X} \mid \phi) = \sum_y \left(P(\overline{X} \mid y, \phi) P(y \mid \phi)\right)$. The substitution of this total probability into our equation gives us: $l(\phi) - l(\phi_n) = \ln\left(\sum_y \left(P(\overline{X} \mid y, \phi) P(y \mid \phi)\right)\right) - \ln\left(P(\overline{X} \mid \phi_n)\right)$. This equation includes the logarithm of a summation. This is where Jensen's Inequality will be utilized (Borman, 2004). In order to apply Jensen's Inequality, constants $\alpha_i \in [0, \infty)$ and $\sum_{i=1}^{n} \alpha_i = 1$ are required. Consider the probability $P\left(y \mid \overline{X}, \phi_n\right)$. Since a probability has a value of at least zero, and the sum of all the elements in the probability is equal to one, therefore by introducing $1 = \frac{P\left(y \mid \overline{X}, \phi_n\right)}{P\left(y \mid \overline{X}, \phi_n\right)}$ into the problem, the requirements to use Jensen's Inequality have been fulfilled. Introducing these constants yields:

$$l(\phi)-l(\phi_n)=\ln\left(\sum_y\left(P(\overline{X}\mid y,\phi)P(y\mid\phi)\frac{P(y\mid\overline{X},\phi_n)}{P(y\mid\overline{X},\phi_n)}\right)\right)-\ln\left(P(\overline{X}\mid\phi_n)\right)$$

$$=\ln\left(\sum_y P(y\mid\overline{X},\phi_n)\frac{P(\overline{X}\mid y,\phi)P(y\mid\phi)}{P(y\mid\overline{X},\phi_n)}\right)-\ln\left(P(\overline{X}\mid\phi_n)\right)$$

$$\geq\sum_y P(y\mid\overline{X},\phi_n)\ln\left(\frac{P(\overline{X}\mid y,\phi)P(y\mid\phi)}{P(y\mid\overline{X},\phi_n)}\right)-\ln\left(P(\overline{X}\mid\phi_n)\right)\quad\text{by Jensen's Inequality}$$

$$=\sum_y P(y\mid\overline{X},\phi_n)\ln\left(\frac{P(\overline{X}\mid y,\phi)P(y\mid\phi)}{P(y\mid\overline{X},\phi_n)P(\overline{X}\mid\phi_n)}\right)$$

and then: $l(\phi)\geq l(\phi_n)+\sum_y P(y\mid\overline{X},\phi_n)\ln\left(\frac{P(\overline{X}\mid y,\phi)P(y\mid\phi)}{P(y\mid\overline{X},\phi_n)P(\overline{X}\mid\phi_n)}\right)$

Next, the M-step must be solved for ϕ. The value we solve for is ultimately represented as ϕ_{n+1}, and since we already know ϕ_n from the previous iteration of the EM Algorithm, ϕ is the only variable that remains.

For convenience, let $Q(\phi\mid\phi_n)=l(\phi_n)+\sum_y P(y\mid\overline{X},\phi_n)\ln\left(\frac{P(\overline{X}\mid y,\phi)P(y\mid\phi)}{P(y\mid\overline{X},\phi_n)P(\overline{X}\mid\phi_n)}\right)$.

Since $l(\phi)$ is an increasing function, and is an upper bound for $Q(\phi\mid\phi_n)$, the EM Algorithm finds ϕ_{n+1} that increases $Q(\phi\mid\phi_n)$ which will also increase $l(\phi)$ at each iteration. Since $Q(\phi\mid\phi_n)$ is a lower bound of our log-likelihood function, finding the maximum of $Q(\phi\mid\phi_n)$ with respect to ϕ is sufficient for the M-step. With a little manipulation, we obtain:

$$\begin{aligned}\phi_{n+1} &= \arg\max_{\phi}\{Q(\phi \mid \phi_n)\} \\ &= \arg\max_{\phi}\left\{ l(\phi_n) + \sum_y P(y \mid \overline{X}, \phi_n)\ln\left(\frac{P(\overline{X} \mid y, \phi)P(y \mid \phi)}{P(y \mid \overline{X}, \phi_n)P(\overline{X} \mid \phi_n)}\right)\right\} \\ &= \arg\max_{\phi}\left\{ \sum_y P(y \mid \overline{X}, \phi_n)\ln\Big(P(\overline{X} \mid y, \phi)P(y \mid \phi)\Big)\right\} \\ &= \arg\max_{\phi}\left\{ \sum_y P(y \mid \overline{X}, \phi_n)\ln\Big(P(\overline{X}, y \mid \phi)\Big)\right\} \\ &= \arg\max_{\phi}\left\{ E_{y\mid\overline{X},\phi_n}\Big\{\ln\Big(P(\overline{X}, y \mid \phi)\Big)\Big\}\right\}\end{aligned}$$

The E-step and the M-step can be repeated multiple times to achieve the best possible maximization as long as some estimation of ϕ is given. The main point is that the maximization of $Q(\phi \mid \phi_n)$ accounts for missing data, where if we were to just use $l(\phi)$, this would not be the case. In the end, the EM Algorithm alternates from E-step that calculates the expected value of an unknown parameter to the M-step that re-estimates the unknown parameter by assuming that the expected value from the E-step is actual data (Dempster et al., 1977). In short, we can describe each step as:

E-Step: Calculate $Q(\phi \mid \phi_n) = E_{y\mid\overline{X},\phi_n}\Big\{\ln\Big(P(\overline{X}, y \mid \phi)\Big)\Big\}$

M-step: Find the ϕ that maximizes $Q(\phi \mid \phi_n)$

After an initial parameter estimation ϕ_n, which is used in the first E-step (when $n = 1$), the EM Algorithm is repeated using the solution of the previous M-step for the new parameter estimation of the next E-step until $Q(\phi \mid \phi_n)$ converges.

When $Q(\phi \mid \phi_n)$ converges to a fixed point, instead of the previous inequality: $l(\phi_{n+1}) \geq Q(\phi_{n+1} \mid \phi_n)$, we now have equality, and the solution has been attained and the iteration of the algorithm can stop. Convergence is guaranteed since each iteration of the algorithm increases the expected value, and a probability has an upper bound and absolute maximum of 1.

The next section goes on to explore the concept of big data. By understanding what big data is and how it works, the relationship between the EM algorithm and big data can be assessed.

3. Big data

Big data is a very new concept even in the technology world today. This revolutionary concept has evolved from the idea that data sets started to become so big that they can no longer fit where people wanted to put them: A single hard drive (Olsen, 2013). The development of big data has only come into being as a result of a few key events that have occurred in the last few years. The invention of what are now data centers in combination with the increased availability, affordability and therefore prevalence of cell phone usage leads to rapid and expansive data production. As a result, lots of data are produced.

While data collection started long before computers were invented, the process was slow, and small in scale. Statistical models were applied to samples of populations because recording and studying an entire population of data was not practical. With the advent of computers, it became much easier to quickly sort and organize data. Even still, early computer work was limited as recording of data had to be done manually and locally on one machine because the ease of sharing data between machines wasn't available yet. However, moving physically recorded and stored data into the digital world was a huge step towards the realization of big data. Recording data was revolutionized when the average consumers started carrying cell phones; more specifically, the smart phone (Olsen, 2013). Prior to the age of cell phones, data recording was not as easy, effortless or automatic as it is today. The digitalization of data combined with electronics automatically recording everything, especially advances in smart phone technology, contributed to the birth of Big data.

As mentioned earlier, the other critical factors that are largely responsible for triggering the evolution of big data were the technological developments that enabled the storage and sharing of huge sets of data. In the early days of computing, manually entering data into a computer was the initial way to record data and store it digitally. Digital data collection, along with sharing and storage used to be limited. Local area networks (LANs) and the Internet were not around when the first personal computers were used for data collection. The size of digital data is labeled the same way that hard drive storage is labeled. Data set sizes, like hard drive sizes are measured in bytes, and each new size category is 1000 times bigger than its predecessor: bytes, kilobytes, megabytes, gigabytes, terabytes, petabytes, etc.. As data were collected and entered onto a single computer, in order to store increasingly more data, one had to build a really big (storage wise, but also physically big depending on how far back in time) supercomputer. These supercomputers or mainframe computers served as the data manager for an entire company. Thankfully, the digital world is always evolving. The dawn of the Internet opened doors with data that at the time, seemed impossible. The Internet combined with shared networking systems created a need for central data storage, leading to the invention and establishment of Data Centers.

Figure 3.1 (below) shows the rapid expansion of digital storage since 1986, which contributed to the demand for data centers (Hilbert, M., & López, P., 2011).

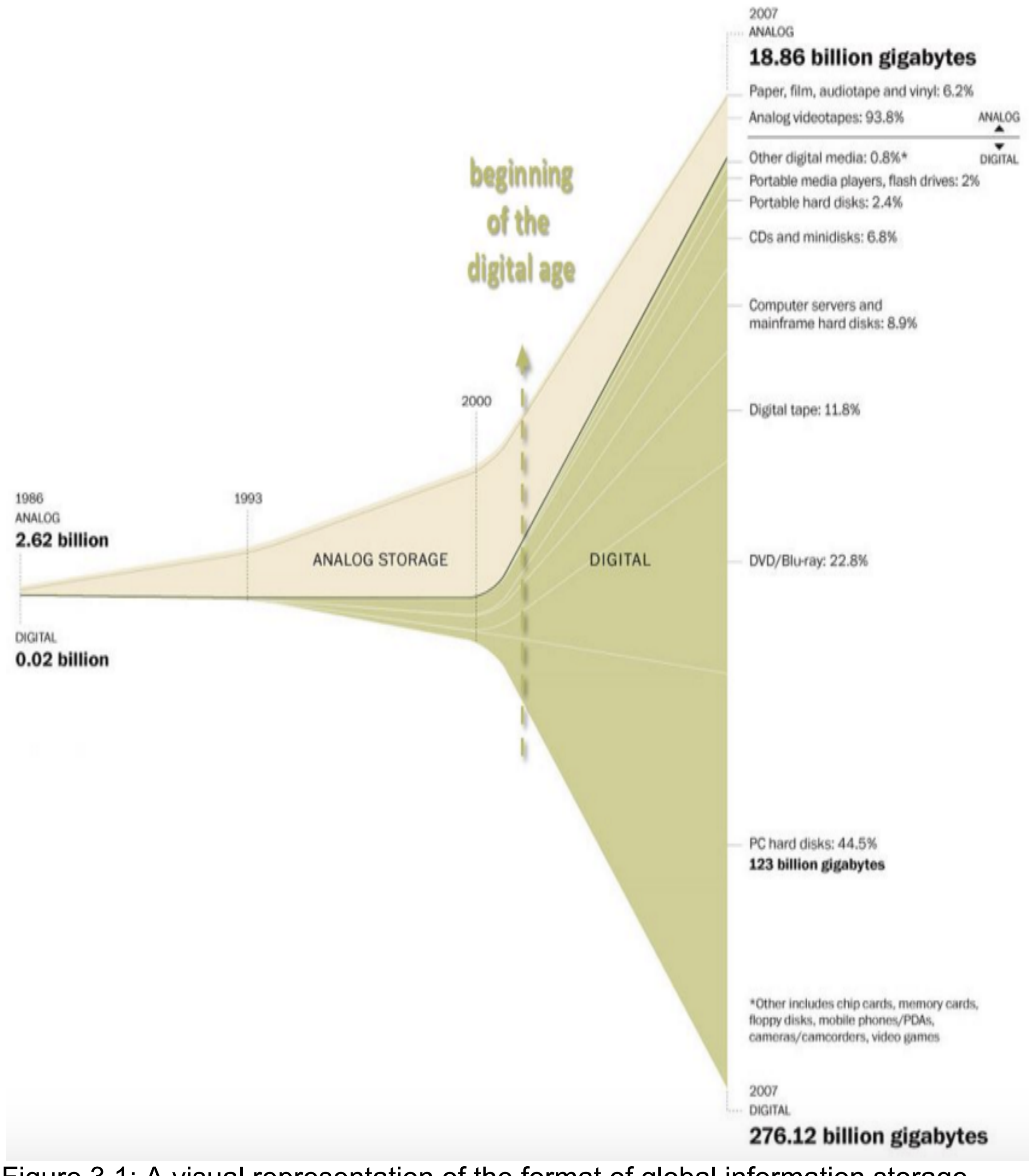


Figure 3.1: A visual representation of the format of global information storage capacity (analog versus digital) from 1986-2007 (Hilbert, M., & López, P., 2011).

The first data centers resulted from early attempts to make a computer that could access a lot of data on a budget. As it turned out, many smaller, less expensive computers can be linked and work together to organize and index data that are stored across multiple machines. Each of the smaller computers had a local hard disk and a local processor. By combining the processing power and storage space of all these smaller machines, the computing power, speed, and efficiency easily surpassed the best supercomputers of the time. This new system of linking computers together is called grid computing (Olsen, 2013; Berstis, 2002). Once grid computing was discovered, the data center was born. Nowadays, data centers can store massive amounts of information, making it possible for companies to store big data. Furthermore, analysts can access the stored big data to study trends and solve problems.

Despite the advances brought about by the invention of data centers, the system remains imperfect. There is a bottleneck on the maximum volume of storage available at data centers; the bottleneck is based on keeping the data center cool, and the ability to increase the digital storage volume while maintaining the physical size of a hard drive or physically decreasing the size hard drive while maintaining digital storage volume. Google has gone to the extreme of demanding and acquiring hardware from its suppliers that dynamically changes its power consumption based on the load of the rest of the parts in the computer (Doctorow, 2008). In data centers, this has had a snowballing effect on cooling. If the power consumption of idle components of hardware can be minimized, and then this is done with every part in the machine, the amount of energy required to keep the parts cool drops drastically. This reduction of energy consumption allows the saved energy to be redirected or saved for another time. By identifying bottlenecks and then taking action to minimize its effect has resulted in Google becoming the technology giant that they are today (Doctorow, 2008).

Continuing the technological evolution, data centers led to the development of cloud computing and cloud data. Cloud data is located in data centers and accessed remotely through a network. Cloud data can be small or large, and most importantly can have multiple users access the data, add to the data, remove data, or edit the cloud data. The cloud has made it so that many people can collect, record, and organize data simultaneously; speeding up the time it would take to access and analyze the results.

The definition of big data was added to the Oxford English Dictionary in 2013, defined as: "data of a very large size, typically to the extent that its manipulation and management present significant logistical challenges." (Press, 2013) Big data are data that won't fit in a typical storage location (Olsen, 2013). The size of a typical storage location is not fixed, as it is changing with the rate that new technology develops. The ability to store and analyze data with the software in these typical storage locations also changes with the rate of technological development. Typical storage locations include but are not limited to personal computers, home servers, cell phones, external hard drives, USB flash drives, etc. These are basic places where the average person can easily access and store data. In 2013, the size of some of the largest consumer hard drives ranged from one to four terabytes, while the quantity of data stored at that time has been estimated to be at least 1,200 exabytes (1 exabyte equals 1 million terabytes). Big data sets can be over 1000 times larger than the storage capacity of consumer hard drives (Lehikoinen & Koistinen, 2014). To aid in the definition of Big data, high volume (quantity of data), high variety, and high velocity (speed of data generation) are generally accepted as the factors that influence the size of big data (Laney, 2001; Olsen, 2013). More recently, veracity (quality of data), variability (inconsistency of data), and complexity have been identified by a number of organizations as characteristics of big data, though everyone hasn't accepted these characteristics, yet.

Due to the growth of raw data, technology, and computing, data that are considered big data today may not be big data 10 years from now. This growth can't be plotted on a graph with a smooth equation. It happens more in bursts, like a step function, depending on new processor technology and storage space available (Doctorow, 2008). If, in a given year, new discoveries in storage technology are made, but processor technology remains on a plateau, the data produced would never be able to fill up the storage space available. In this instance, the demand to produce data faster would increase. If, processor advances were made without increasing the available storage space, data would be produced so fast that it could not be stored. Thus, there would be a demand to increase storage space to fit the new data. The growth bursts in data and computing occur when advancement happens in both storage and processor technology (Doctorow, 2008).

Even though it can be relatively easy to obtain or generate big data, this does not imply Big data is also automatically sorted, organized, or understood the moment it is recorded. With a large data set, there can be a substantial part of the data that is completely useless. Since these useless parts take up so much space, filters have been a way to reduce the volume of information being recorded. This can ease the process of sorting and organizing the data but brings up the challenge of making sure not to discard the useful parts of a big data set.

The practical applications of analysis of big data are seen in something as simple as the auto fill feature when typing something into a Google search bar. The search engine automatically gives options to what is being typed based on the big data set that consists of all searches typed. By using big data, queries can be answered more efficiently. This example of search engine data can be analyzed with Topology to contribute to the auto-fill option that shows up when typing a word or phrase into the search bar. In the next section, we will explore what happens when Topology is used in tandem with Data Analysis in the context of big data.

4. Topological Data Analysis

Computational or Algebraic Topology and Data Analysis can be combined to describe big data. This description can be used to find patterns in the data, obtain knowledge, and solve problems. One may ask: Why combine topology and data analysis? The benefit of combining topology with data analysis is that one of the fundamental parts of topology is reducing sets of points to their qualitative features. These qualitative features define different topologies that can be used to describe and categorize the sets of points. An important goal of data analysis is to gain knowledge of data (understand large scale organization of the data), and by combining data analysis with topology, obtaining the needed qualitative information is much easier (Carlsson, 2009). This information is usually higher than 3-dimensional, making it difficult to visualize. However, these qualitative features can be translated into various images and also allow for a visual understanding of the data. This is done using topology to project data that are n-dimensional (for $n \geq 3$) to a 3-dimensional space is shown in Figure 4.6.

4.1. Topology

Topology is a subject of mathematics that uses qualitative geometric properties of sets. For example, Topology is concerned whether or not sample spaces are connected, which can lead to classification of loops within the spaces. Topology ignores the qualitative features of distance metrics, and instead uses the property of connectedness to declare that all points in the same connected space are infinitely close or infinitely far from each other. This means that two shapes are topologically equivalent if you can take one shape and stretch, compress, drag, etc. so that it looks the same as the second shape. If at any point a disconnection or separation of parts of the shape must occur in order to achieve equivalence, then the two shapes are not in the same topology. As a simple example, consider the following capital letters from the alphabet:

A B C D E

Assume that the only points in each set are the ones that make the letter. With each letter representing one set of points, there are 5 different sets, and all look visually different. Topologically, there are two pairs of equivalent sets and one set that is unique in the sample. The fastest way to classify each letter in this sample is to count how many loops are required to compose the letter. After the loops, the rest can be dragged, stretched or compressed to look like another letter. C & E both have zero loops. A & D both have one loop. B is unique in this sample with two loops.

In topology, these equivalence classes of loops are called a homotopy group. To understand this the following definitions are provided according to *Topology, 2nd Edition* (Munkres, 2000):

Definition: Homotopy

If f and g are continuous maps of the space X into the space Y, with $I=[0,1]$ we say f is homotopic to g if there is a continuous map $F: X\times I \to Y$ such that $F(x,0)=f(x)$ and $F(x,1)=g(x)$ for each x. The map F is called a homotopy between f and g. If f is homotopic to g, then we write $f \simeq g$ (Munkres, 2000).

Definition; Path

If $f:[0,1]\to X$ is a continuous map such that $f(0)=x_0$ and $f(1)=x_1$, then f is a path in X from x_0 to x_1 (Munkres, 2000).

Definition: Path Homotopy

If two paths f and g map the interval $I=[0,1]$ into X, and if they have the same initial point x_0 and the same terminal point x_1, and if there is a continuous map $F: I\times I\to X$ such that

$$\begin{aligned} F(s,0)&=f(s) \quad \text{and} \quad F(s,1)=g(s),\\ F(0,t)&=x_0 \quad\ \ \text{and} \quad F(1,t)=x_1, \end{aligned}$$

for each $s\in I$ and each $t\in I$, then f is path homotopic to g and F is a path homotopy. If f is path homotopic to g, then we write $f\simeq_p g$ (Munkres, 2000).

It is important to note that the relations in both Homotopy and Path Homotopy are equivalence relations, so this allows us to look at path homotopies as equivalence classes.

Definition: Product of Paths

If f is a path in X from x_0 to x_1, and if g is a path in X from x_1 to x_2, the product $f * g$ of f and g is defined with the path h given from the following equations:

$$h(s) = \begin{cases} f(2s) & \text{for } s \in [0, \frac{1}{2}], \\ g(2s-1) & \text{for } s \in [\frac{1}{2}, 1]. \end{cases}$$

The operation $*$ used in the definition of product of paths allows connected paths to be combined into a well-defined and continuous function using the pasting lemma (Munkres, 2000).

Definition: Loop

Let X be a space; let x_0 be a point in X. A path in X that begins and ends at x_0 is a loop based at x_0 (Munkres, 2000).

Two loops that are based at the same initial point can be connected using the product of paths. This allows us to put loops into equivalence classes. This allows us to use all of these previous definitions as parts in a complete definition called a Homotopy Group.

Definition: Homotopy Group

The set of path homotopy classes of loops based at x_0, with the operation $*$, is a Homotopy Group (Munkres, 2000).

4.2. Topology and Data Shapes

Topological data analysis (TDA) looks at the shape that data creates. The shape of the data is important. The basis of this is seen on a simple scatter plot. The following figures help create a visual picture of what the goal of TDA is and why it is important to use on data (Knudson, 2015).

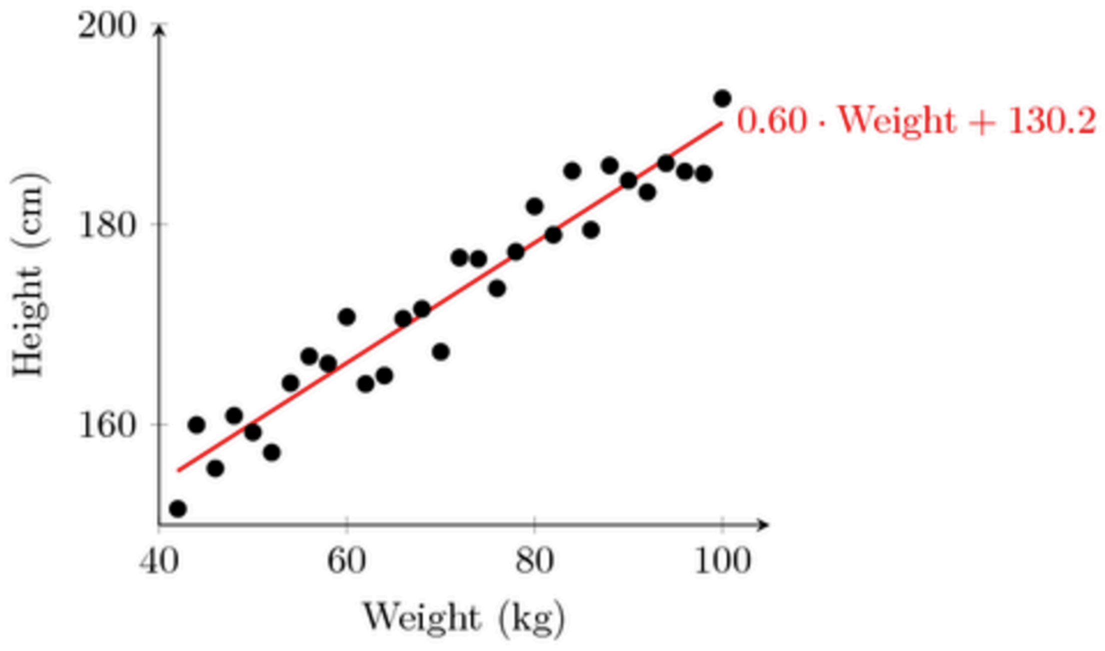


Figure 4.1: An example of data that fits a regression line (Knudson, 2015).

This shape would be the simplest that data could produce. For most people, this is the only way they learned if there was a relationship between two characteristics of data. However, even when using just a scatter plot, it is possible to have more than just a linear relationship. Consider a data set in Figure 4.2 below (Carlsson, 2015, January 6).

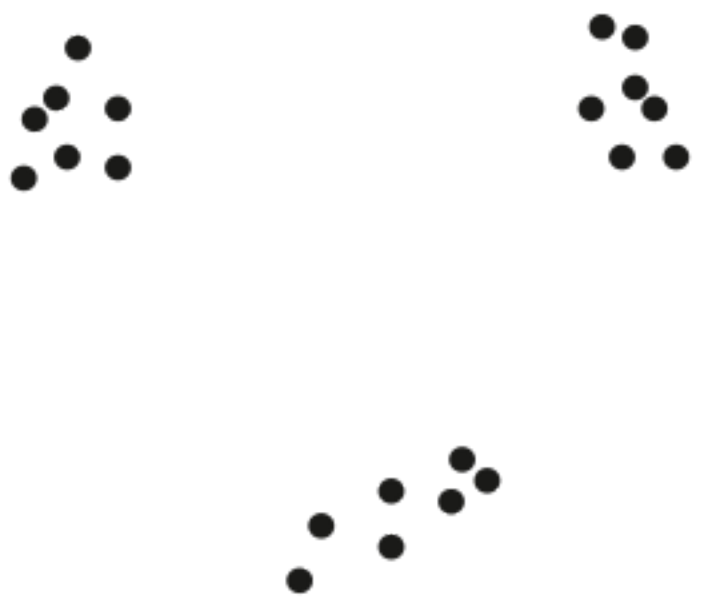


Figure 4.2: A data set that is divided into clusters (Carlsson, 2015, January 6).

There is not a clear way to draw a regression line for these clusters. The data are predictable in that they gravitate towards three points, but not in a way that a regression line equation can predict the location of future data. Figure 4.3 is another example of data that has a shape, but is not a line or clustered (Carlsson, 2015, January 6).

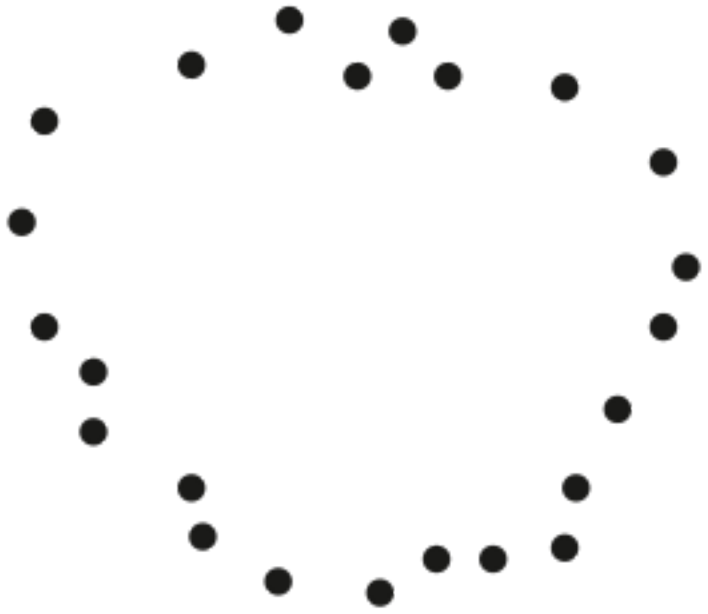


Figure 4.3: Data that are arranged in a circular shape (Carlsson, 2015, January 6).

This circular shape is connected to a repeating tread, or recurring pattern in the data, much like familiar trigonometric equations; sine and cosine. The data in Figure 4.3 could be approximated using a form of the unit circle: $\sin^2(t) + \cos^2(t) = 1$, due to other factors (known or unknown) influencing how close the approximation would be. Figure 4.4 is another example of data that has a clear shape (Carlsson, 2015, January 6).

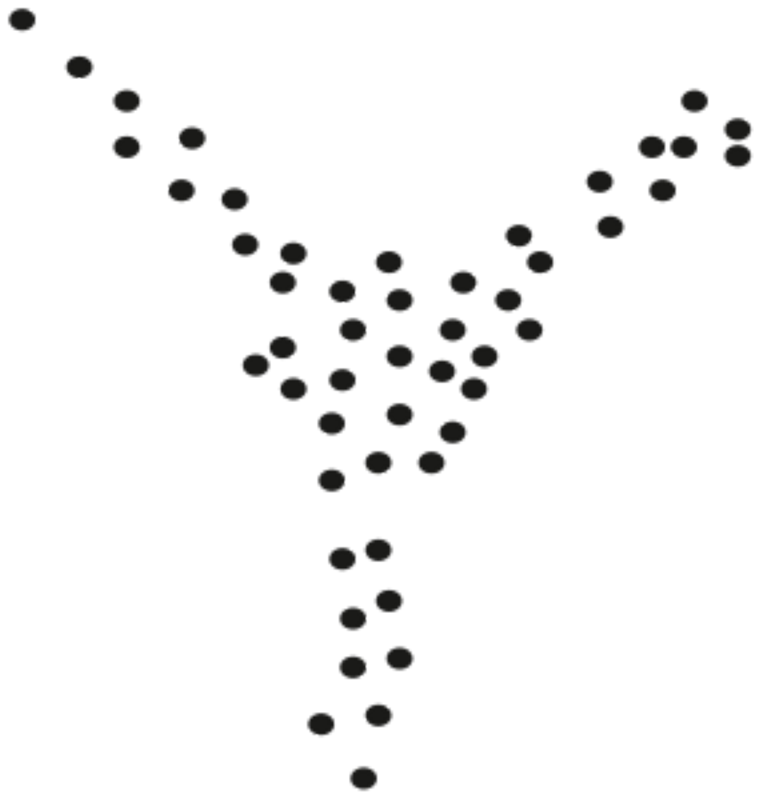


Figure 4.4: Data that are shaped like a “Y” (Carlsson, 2015, January 6).

Even though the data seem to be clumped around the center, there are three obvious extensions that could be related to some extreme characteristic. This could also be a result again of periodic behavior, and since each data point could have been collected at a different point in time, it creates a loopy shape kind of like a “Y.” The interesting thing about this data set is we can probably describe it really well if we refer to polar graphing. This shape is similar to the polar equation $r = \sin(3\theta)$, seen in figure 4.5.

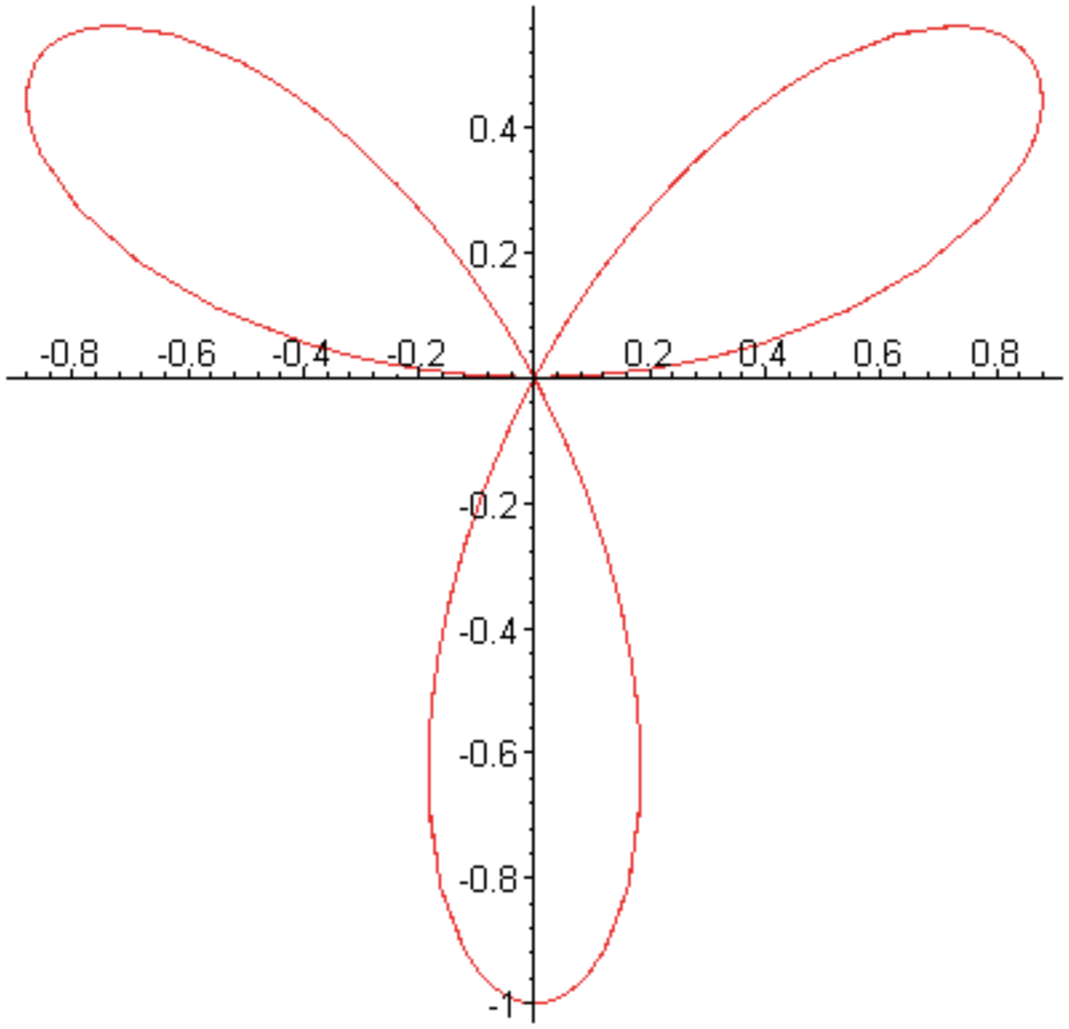


Figure 4.5: The polar graph of the equation $r = \sin(3\theta)$. This graph doesn't match Figure 4.4 exactly, but similarity exists. It is obvious that data creating a shape is important and is used to describe data to contribute to finding more information about the data.

Each of these example Figures 4.2, 4.3, & 4.4 of visualized data sets are distinct in shape, and all differ from the linear example in Figure 4.1 with the regression line. These examples can be seen in real data sets, and are just simple examples of patterns or shapes that can show up in data. Most data is not as simple as these figures show. Big data can be very complex due to its size. TDA works wonderfully on organizing and aiding in interpreting complex data. Complex data can include all of the simple examples mentioned above, as well as many other shapes that were not mentioned (Carlsson, 2015, March 23). That is what makes TDA so important with big data.

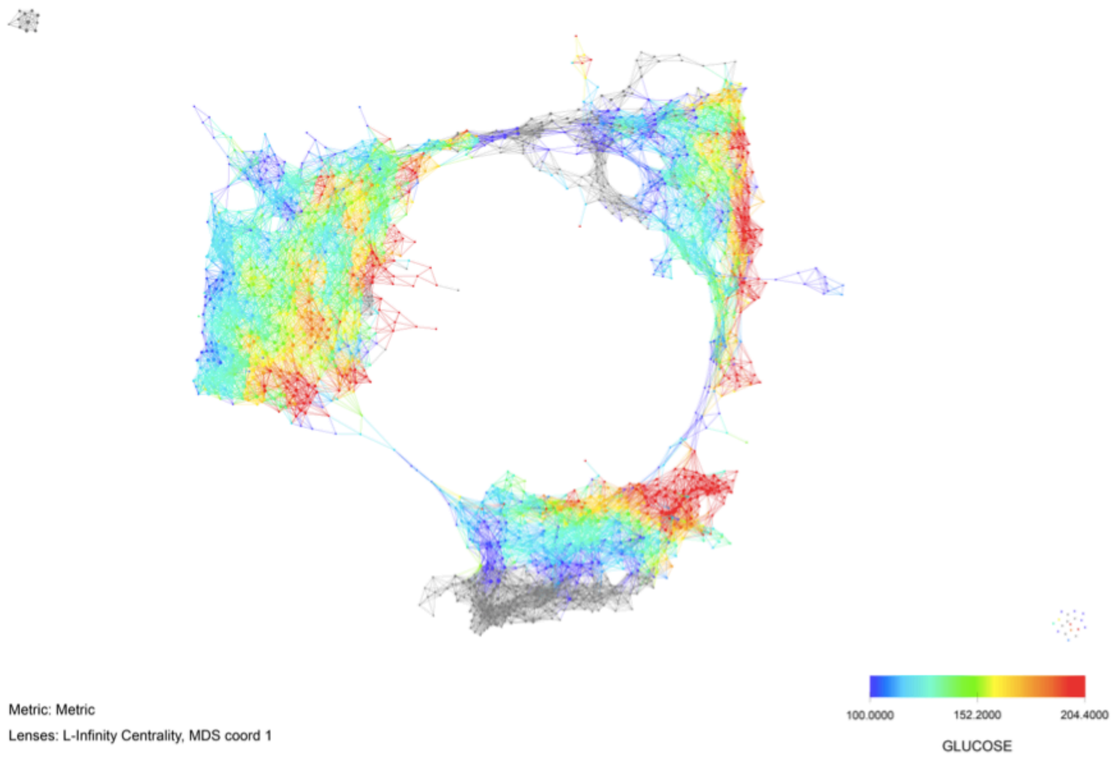


Figure 4.6: An example of complex data (Carlsson, 2015, March 23). Not all data are easy to work with by fitting onto a straight line or into a plane. These complex data are difficult to visualize, because the data are plotted in a 3-dimensional space, when the figure can only capture two of the dimensions. Representing data in a topological network allows one to see relationships in data that would otherwise go unseen because taking data and plotting it on a two

dimensional X-Y axis doesn’t promise easy clumping into following regression lines as seen in the previous figures.

Figure 4.6 was created using Ayasdi Core, an analytics software program that was created for using TDA on data. This software was designed to create 3-dimensional topological networks like Figure 4.6, and add color to the image in order to add a fourth dimension to the figure, enhancing the ability for the viewer to interpret relationships among the clusters of data (Long, 2015).

4.3 EM Algorithm and Big data

In 2008, Wolf et al. completed a fully distributed EM Algorithm for very large datasets (big data). Their conclusion was that both in theory, and in actuality, using the EM Algorithm on big data could be done. They describe a few network topologies that were used in their distribution of the EM Algorithm on the very large datasets. The general presentation of the paper was to talk about the different results of using different topologies, and the effect each topology has on communication bandwidth, communication latency, and per node memory requirements (Wolfe et al., 2008). Some of their results include the ability for a large speed increase of the EM Algorithm and computations scaling up to be too large for a single machine. Even though the computations would be large, using grid computing would allow these computations to be completed easily compared to attempts on a single machine.

In using the EM Algorithm on big data, Wolfe used three different topologies to see which one allowed for the EM Algorithm to be completed the fastest: MapReduce Topology, AllPairs Topology, and JunctionTree Topology (Wolfe et al., 2008). To do this, the data must be broken down into clusters of data, and apply the EM Algorithm to each data cluster. After that, the topology used will define how the information from the clusters is used to apply the EM Algorithm to get the result from the entire dataset.

5. Conclusion

In this essay, it is demonstrated that the EM Algorithm can be used on big data. The first step, the e-step (expectation step) is relatively easy to complete, but the m-step (maximization step) is much more complicated with big data because of its size. Big data has created data samples that behave more like a population of data compared to the samples most statisticians have dealt with. Big data sets are a relatively recent phenomena, as the advent of data centers and personal cell (smart) phones created the ability to generate data in volumes unheard of in prior periods. The ability to generate increasingly larger data sets is triggered and driven by technology advancements. Both Processor speed improvements and storage density improvements are required to enlarge big data sets, as only one technology improvement without the other merely forces a constraint, or bottleneck, to the other technology. As a result, these data set growths appear in spurts, rather than in a linear fashion. However, the data collected are applicable to many circumstances and are used in our daily lives. The example chosen here is the use of the “autofill” function in Google’s search engine.

However, big data sets need to be analyzed to be useful, and historically popular algorithms and analysis tools are defined for samples from a population of data. These tools tend to be insufficient to provide meaningful analysis. Due to the size of big data, help in handling the size to make the calculation is necessary. Several examples are shown illustrating this fact, as is a discussion of Topological Data Analysis (TDA) to provide qualitative descriptions of big data. TDA reduces big data to sets of points, or discrete data, into qualitative features that represent the data, or important features of the data, as a whole.

Although the growth of the size of big data is difficult to predict, the current environment is one that is in ever-increasing need for tools to assist the analysis of these data. Therefore, as demonstrated earlier, using topology (and more specifically, TDA) on a big data set is the main vehicle that makes the application of the EM algorithm onto big data possible. Pairing the advancement of technology with TDA has opened the keys to doors in mathematics that previously seemed to be impossible to open.

References

Berstis, V. (2002). Fundamentals of Grid Computing (pp. 1-28): IBM Corp.

Borman, S. (2004). The Expectation Maximization Algorithm - A short tutorial.

Carlsson, G. (2009). Topology and Data. Bulletin of the American Mathematical Society, 46(2), 255-308. Retrieved from, http://www.ams.org/journals/bull/2009-46-02/S0273-0979-09-01249-X/S0273-0979-09-01249-X.pdf

Carlsson, G. (2015, January 6). Why Topological Data Analysis Works | Ayasdi. Retrieved from, http://www.ayasdi.com/blog/bigdata/why-topological-data-analysis-works/

Carlsson, G. (2015, March 23). What Exactly is Complex Data? | Ayasdi. Retrieved from, http://www.ayasdi.com/blog/topology/exactly-complex-data/

Dempster, A. P., Laird, N. M., & Rubin, D. B. (1977). Maximum Likelihood from Incomplete Data via the EM Algorithm. *Journal of the Royal Statistical Society: Series B, 39*(1), 1-38.

Doctorow, C. (2008) *The Challenges of Big data*.

Fayyad, U., Piatesky-Shapiro, G., & Smyth, P., (1996) From Data Mining to Knowledge Discovery in Databases. *AI Magazine,* Fall 1996: 37-54. Retrieved from http://www.kdnuggets.com/gpspubs/aimag-kdd-overview-1996-Fayyad.pdf

Harris, J. W., & Stocker, H. (1998). *Handbook of Mathematics and Computational Science*. New York: Springer-Verlag.

Hilbert, M., & López, P. (2011). The World's Technological Capacity to Store, Communicate, and Compute Information. *Science, 332*(6025), 60 –65. doi:10.1126/science.1200970

Knudson, K. (2015, May 26) Big Topology. Retrieved from, http://hplusmagazine.com/2015/05/26/big-topology/

Kohavi, R., Provost, F. (1998). Glossary of Terms. *Machine Learning*, 30: 271-274. Retrieved from http://ai.stanford.edu/~ronnyk/glossary.html

Laney, D. (2001, February 6) 3D Data management: Controlling Data Volume, Velocity, and Variety. Gartner. Retrieved from, http://blogs.gartner.com/doug-laney/files/2012/01/ad949-3D-Data-Management-Controlling-Data-Volume-Velocity-and-Variety.pdf

Lehikoinen, J., & Koistinen, V. (2014). In Big data We Trust? *Interactions*, 38-41. doi: 10.1145/2641398

Long, J. (2015, April 23). Baseball Prospectus | Pitching Backward: Every Player in Its Right Place. Retrieved from, http://www.baseballprospectus.com/article.php?articleid=26142#

Munkres, J. (2000). *Topology* (2nd ed.) Upper Saddle River, NJ: Prentice Hall.

Olsen, M. (2013) *Opportunities Abound in the Big data Space/Interviewer: P. Li*.

Press, G. (2013, June 13). Big data News: A Revolution Indeed. Retrieved from, http://www.forbes.com/sites/gilpress/2013/06/18/big-data-news-a-revolution-indeed/

Rogers, P. (2015, March 6). The Big data Pivot is in Full Swing | Ayasdi. Retrieved from, http://www.ayasdi.com/blog/bigdata/big-data-pivot-full-swing/

Weisstein, E. W. Likelihood Function. (2015) *MathWorld -- A Wolfram Web Resource.* from http://mathworld.wolfram.com/LikelihoodFunction.html

Wolfe, J., Haghighi, A., & Klein, D. (2008). Fully Distributed EM for Very Large Data Sets.