Introduction

Jianxin Wu

LAMDA Group
National Key Lab for Novel Software Technology
Nanjing University, China
wujx2001@gmail.com

February 18, 2020

Contents

This book will introduce several algorithms, methods, and practices in *pattern recognition*, with a sizable portion of its contents being the introduction to various *machine learning* algorithms.

Technical details (such as algorithms and practices) are important. However, we encourage the readers to focus on more fundamental issues rather than dwelling on technical details. For example, keeping the following questions in mind will be useful.

- What is pattern recognition? What is machine learning? And, what is the relationship between them?
- Given a specific pattern recognition task, what is the input and output of this task? What aspects of the task make it difficult to solve?
- Among the many existing algorithms and methods, which one(s) should be used (or must not be used) for our task at hand? Is there any good reason to support your decision?

- There is no silver bullet that can solve all problems. For example, deep learning has emerged as the best solution technique for many applications, but there are many tasks that cannot be effectively attacked by deep learning. If we have to develop a new method for one task, is there a procedure that can help us?

Although we have not defined these terms—such as pattern recognition and machine learning—it is worthwhile to quickly scan these questions. In fact, there is no crystal clear answer to these questions. For some (e.g., the last two), researchers and practitioners mostly resort to their experiences in choosing a particular attack for their task at hand. An experienced researcher may pick an appropriate algorithm for his or her task in the first trial. A novice, however, may try all the methods from a handbook in a random order, which means a lot of time, energy, and cost is often wasted before the first working method emerges for this task. Hence, rather than remembering every technical detail of the methods we introduce, it is probably more important to build your own rule of thumb for the following question through our introduction and analyses of these methods: *what is this method good for (or bad for)?*

To answer some other questions, we require knowledge beyond the materials we will introduce in this book. For example, in order to understand what are the input, output, and difficulties in OCR (optical character recognition, a typical pattern recognition task), we need to carefully study at least the specific environment the OCR task will happen in and the level of recognition accuracy that is required. All these factors are beyond the scope of a brief introduction to pattern recognition and machine learning. Although it is impossible to dive into such application specific details, we will remind the readers about the importance of these factors when appropriate.

There remain questions that have very different answers from different communities or individuals. For example, what is pattern recognition? What is machine learning? What is the relationship between them?

Let us take the widely utilized Wikipedia project as an example.[1] The entry "Pattern recognition" was introduced as follows in Wikipedia as of Oct 8, 2016, which also involves machine learning, and reflects at least one type of understanding of this subject.

> Pattern recognition is a branch of machine learning that focuses on the recognition of patterns and regularities in data, although it is in some cases considered to be nearly synonymous with machine learning.
>
> – "Pattern recognition," Wikipedia, retrieved on Oct 8, 2016.

Along with our introduction of both subjects, however, we want to emphasize that although machine learning (ML) and pattern recognition (PR) are closely related subjects, *PR is* not *a branch of ML; neither is ML a branch of PR.*

In this book, we adopt another definition of pattern recognition:

[1] https://en.wikipedia.org/wiki/Main_Page

> The field of pattern recognition is concerned with the automatic discovery of regularities in data through the use of computer algorithms and with the use of these regularities to take actions such as classifying the data into different categories.
>
> – "Pattern Recognition and Machine Learning," Christopher M. Bishop

Pattern recognition is closely related to machine learning, which is defined as follows.

> The field of machine learning is concerned with the question of how to construct computer programs that automatically improve with experience.
>
> – "Machine Learning," Tom Mitchell

These definitions may seem difficult to grasp at first sight. Let us start with an example that are related to both of them.

1 An example: autonomous driving

We will use autonomous driving as an example to illustrate machine learning, pattern recognition, and other subjects.

Ideally, a fully autonomous car (which is not yet commercially available as of today) may operate as follows.

`T.1` The car identifies its owner is approaching and automatically unlocks its doors.

`T.2` The car will communicate with the user to learn about the destination of a trip.

`T.3` The car will find its way and drive to the destination on its own (i.e., autonomously), while the user may take a nap or enjoy a movie during the trip.

`T.4` The car will properly notify the user upon its arrival and shall park itself after the user leaves the car.

For task `T.1` above, a viable approach is to install several cameras in different sides of the car, which can operate even when the engine is switched off. These cameras will watch out vigilantly and automatically find out that a person is approaching it. Then, the next step is to identify the person: is he or she the owner of this car? If the answer is yes and when the owner is close enough to any one of the car's doors, the car shall unlock that door.

Because these steps are based on cameras, the task `T.1` can be viewed as a *computer vision* (CV) task.

Computer vision methods and systems take as inputs the images or videos captured by various imaging sensors (such as optic, ultrasound, or infrared cameras). The goal of computer vision is to design hardware and software that can work together and mimic or even surpass the functionality of the human vision system—e.g., the detection and recognition of objects (such as identifying a person) and the identification of anomalies in the environment (such as finding that a person is approaching). In short, the input of computer vision methods or systems are different types of images or videos, and the outputs are results of image or video understanding, which also appear in different forms according to the task at hand.

Many subtasks have been decomposed from `T.1`, which correspond to widely researched topics in computer vision, e.g., pedestrian detection and human identity recognition (such as face recognition based on human faces, or gait recognition based on a person's walking style and habits).

Task `T.2` involves different sensors and data acquisition methods. Although we can ask the user to use a keyboard to key in his or her destination, the more natural way is to carry out this communication through natural languages. Hence, microphones and speakers should be installed around the car's interior. The user may just say "Intercontinental hotel" (in English) or "Zhou Ji Jiu Dian" (in Chinese). It is the car's job to find out that a command has been issued and to acknowledge (probably also to confirm) the user's command before it starts driving. The acknowledgment or confirmation, of course, are given in natural languages, too.

Techniques from various areas are required to carry out these natural verbal communications, such as *speech recognition*, *natural language processing*, and *speech synthesis*. The car will capture the words, phrases, or sentences spoken by the user through speech recognition, shall understand their meanings, and must be able to choose appropriate answers through natural language processing; finally it can speak its reply through speech synthesis.

`T.2` involves two closely related subjects: speech processing and natural language processing. The input of `T.2` is speech signals obtained via one or several microphones, which will undergo several layers of processing: the microphones' electronic signal is firstly converted into meaningful words, phrases, or sentences by speech recognition; the natural language processing module will convert these words into representations that a computer can understand; natural language processing is also responsible for choosing an appropriate answer (e.g., a confirmation or a further clarification request) in the form of one or several sentences in texts; finally, the speech synthesis module must convert the text sentences into audio signals, which will be spoken to the user via a loudspeaker and is the final output of `T.2`. However, the modules used in `T.2` have different intermediate inputs and outputs, often with one module's output being the input of the next processing module.

`T.3` and `T.4` can be analyzed in a similar fashion. However, we will leave the analyses of `T.3` and `T.4` to the reader.

In this example, we have witnessed many sensors (e.g., camera, infrared camera, microphone) and outputs of various modules (e.g., existence of a person,

human identity, human voice signal). Many more sensory inputs and outputs are required for this application. For example, in `T.3`, highly accurate global positioning sensors (such as GPS or BeiDou receiving sensors) are necessary in order to know the car's precise location. Radars, which could be millimeter-wave radar or laser based (Lidar), are also critical to sense the environment for driving safety. New parking lots may be equipped with RFID (radio-frequency identification) tags and other auxiliary sensors to help the automatic parking task.

Similarly, more modules are required to process these new sensory data and produce good outputs. For example, one module may take both the camera and radar inputs to determine whether it is safe to drive forward. It will detect any obstacle that stands in front of the car and to avoid collision if obstacles do exist.

2 Pattern recognition & machine learning

The above examples are all examples of *pattern recognition*, which automatically extracts useful patterns from input data (e.g., pedestrians from images or texts from speech signals), and the extracted patterns are often used in decision making (e.g., whether to unlock the car or to find an appropriate response to a user's speech command).

The word *pattern* can refer to a wide range of useful and organized information in diverse applications. Some researchers use the word *regularity* as a synonym for the word pattern, both referring to information or knowledge that is useful for future decision making. The word *automatic* refers to the fact that a pattern recognition method or system acts on its own (i.e., without a human in the loop).

2.1 A typical PR pipeline

There are many options to deal with a pattern recognition task. However, the steps in Figure 1 form a typical pattern recognition (PR) pipeline.

A PR pipeline often starts from its input data, which most probably come from various sensors. Sensors (such as cameras and microphones) collect input signals from the environment in which the PR system operates. This step, also termed *data acquisition*, is extremely important for pattern recognition performance.

The properties of input data can be more important than other steps or components in the PR pipeline. Face recognition from surveillance videos is an example to illustrate this statement. Surveillance videos are useful tools in the investigation of accidents or crimes. However, it is rare that surveillance cameras happen to be within small distances to where the accidents or crimes happened. When the cameras are far away from the site (e.g., more than 30 meters away), a face will occupy less than 20×20 pixels in the video frames. This resolution is too small to provide useful identity information for a suspect, either for a human

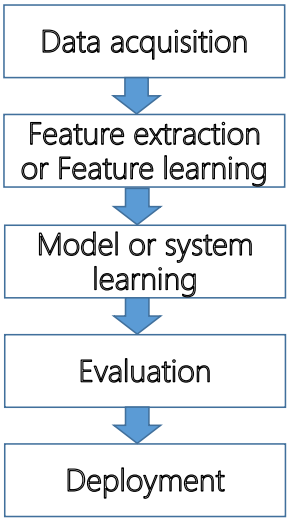


Figure 1: A typical pattern recognition pipeline.

expert or an automatic computer vision or pattern recognition system. Hence, acquiring high quality input data is the top priority in achieving a successful pattern recognition system. If a high resolution facial image (e.g., with more than 300×300 pixels) is available, the face recognition task will become much easier.

Acquiring high quality sensory input involves experts from many fields, for example, physicists, acoustic engineers, electrical and electronics engineers, optical engineers, etc. Sensory input data often need to be digitized, and may appear in many different forms such as text, images, videos, audio signals, or 3D point clouds. Beyond input data directly captured by various sensors, a PR method or system can also use the output of other methods or systems in the PR pipeline as its input.

The next step is feature extraction or feature learning. The raw sensory input, even after digitization, are often far from being meaningful or interpretable. For example, a color picture with resolution 1024×768 and three channels (RGB) is digitized as

$$3 \times 1024 \times 768 = 2\,359\,296$$

integers between 0 and 255. This large quantity of integers is not very helpful for finding useful patterns or regularities in the image.

Figure 2 shows a small gray-scale (single channel) face image (Figure 2a) and its raw input format (Figure 2b) as a matrix of integers. As shown by Figure 2a, although the resolution is small (23×18, which is typical for faces in surveillance videos), our brain can still interpret it as a candidate for a human face, but it is almost hopeless to guess the identity of this face. The computer, however, sees a small 23×18 matrix of integers, as shown in Figure 2b. These

414 numbers are far away from our concept of a human face.

Hence, we need to extract or learn *features*—i.e., to turn these 414 numbers into other numerical values which are useful for finding faces. For example, because most eyes are darker than other facial regions, we can compute the sum of pixel intensity values in the top half image (12×18 pixels, denoting the sum as v_1) and the sum of pixel intensity values in the bottom half (12×18 pixels, denoting the sum as v_2). Then, the value $v_1 - v_2$ can be treated as a feature value: if $v_1 - v_2 < 0$, the upper half is darker and this small image is possibly a face. In other words, the feature $v_1 - v_2$ is useful in determining whether this image is a face or not.

Of course, this single feature is very weak; we may extract many feature values from an input image, and these feature values form a *feature vector*.

In the above example, the features are manually designed. Manually designed features often follow advice from domain experts. Suppose the PR task is to judge whether a patient has a certain type of bone injury, based on a CT image. A domain expert (e.g., a doctor specialized in bone injuries) will explain how he or she reaches a conclusion; a pattern recognition specialist will try to capture the essence of the expert's decision making process and turn this knowledge into feature extraction guidelines.

Recently, especially after the popularization of *deep learning* methods, feature extraction has been replaced by feature learning in many applications. Given *enough* raw input data (e.g., images as matrices) and their associated labels (e.g., face or non-face), a learning algorithm can use the raw input data and their associated labels to automatically learn good features using sophisticated techniques.

After the feature extraction or feature learning step, we need to produce a model, which takes the feature vectors as its input, and produces our application's desired output. The model is mostly obtained by applying *machine learning* methods on the provided training feature vectors and labels.

For example, if an image is represented as a d-dimensional feature vector $\boldsymbol{x} \in \mathbb{R}^d$ (i.e., with d feature values), a linear model

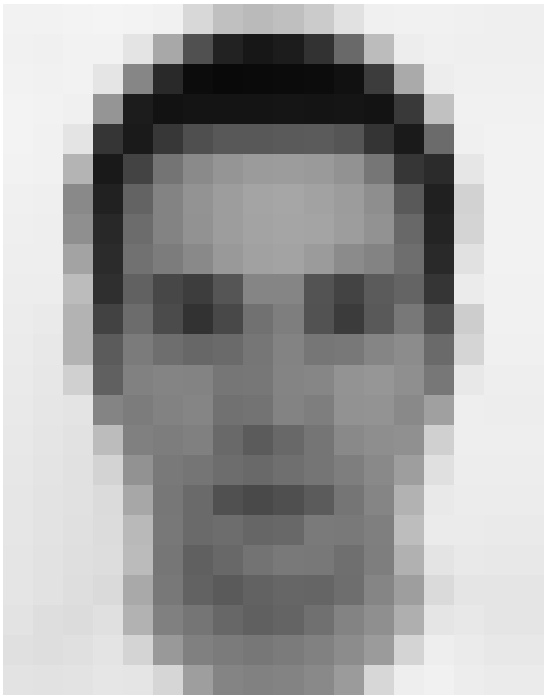
$$\boldsymbol{w}^T \boldsymbol{x} + b$$

can be used to produce the output or prediction, in which

$$\boldsymbol{w} \in \mathbb{R}^d \text{ and } b \in \mathbb{R}$$

are $d + 1$ *parameters* of the linear machine learning model. Given any image with a feature vector $\boldsymbol{x}$, the model will predict the image as being a face image if $\boldsymbol{w}^T \boldsymbol{x} + b \geq 0$, or non-face if $\boldsymbol{w}^T \boldsymbol{x} + b < 0$.

In this particular form of machine learning model (which is a parametric model), to learn a model is to find its optimal parameter values. Given a set of training examples with feature vectors and labels, machine learning techniques learn the model based on these training examples—i.e., using past experience (training instances and their labels) to learn a model that can predict for future examples even if they are not observed during the learning process.



(a) A small gray-scale face image

242	242	243	244	243	241	216	192	186	191	203	225	240	241	241	240	239	239
241	240	242	242	228	168	80	34	23	28	50	105	188	237	240	239	239	239
242	241	242	229	134	41	12	9	10	11	12	17	60	170	237	240	240	240
243	242	240	149	28	18	20	20	20	21	20	21	19	57	193	241	241	241
243	241	228	53	26	55	79	88	88	90	91	85	52	26	107	240	242	242
242	241	180	25	67	116	138	149	155	156	154	146	118	54	43	230	242	242
240	239	137	33	99	131	147	157	164	166	162	156	142	89	32	210	242	242
239	238	143	40	108	131	146	157	163	165	164	157	146	104	36	212	242	242
238	238	163	43	113	124	137	153	161	163	155	139	131	110	41	225	242	242
237	237	188	47	97	71	61	85	133	133	83	67	91	100	53	235	241	241
236	234	179	66	108	75	50	72	113	126	82	59	89	120	79	205	241	241
235	232	179	91	123	110	103	106	118	131	118	120	133	140	106	213	240	240
234	231	208	96	129	132	131	118	119	132	134	147	150	143	119	232	238	238
233	230	228	132	124	130	133	113	115	131	126	146	147	137	160	238	237	237
232	229	226	188	128	125	128	105	91	104	117	137	142	144	208	238	238	238
231	228	225	211	146	121	117	109	105	111	117	126	137	158	224	237	237	237
230	227	225	219	167	119	106	80	73	79	91	117	133	178	233	236	236	236
230	227	224	220	185	120	106	109	102	104	122	121	127	189	235	235	234	234
229	226	223	221	187	118	96	104	115	121	120	111	126	177	228	233	232	232
229	226	222	218	169	120	98	90	98	102	101	110	133	158	218	229	229	229
227	222	220	224	178	128	117	103	96	100	112	131	143	175	229	232	229	229
224	222	224	228	212	153	128	123	119	123	129	140	155	212	238	235	232	232
227	226	227	230	229	214	158	132	129	131	137	155	215	238	241	237	235	235

(b) The image seen by a computer

Figure 2: A small gray-scale image (23×18 in resolution, and enlarged by 15 times) in Figure 2a. It is seen by the computer as a 23×18 matrix of integers, as shown in Figure 2b.

2.2 PR vs. ML

Now we will have a short detour to discuss the relationship between PR and ML before we proceed to the next step in the PR pipeline.

It is quite easy to figure out that pattern recognition and machine learning are two closely related subjects. An important step (i.e., model learning) in PR is typically considered as an ML task, and feature learning (also called representation learning) has attracted increasing attention in the ML community.

However, PR includes more than those components that are ML-related. As discussed above, data acquisition, which is traditionally not related to machine learning, is ultra-important for the success of a PR system. If a PR system accepts input data that is low quality, it is very difficult, if not impossible, for the machine learning related PR components to recover from the loss of information incurred by the low quality input data. As the example in Figure 2 illustrates, a low resolution face image makes face recognition almost impossible, regardless of what advanced machine learning methods are employed to handle these images.

Traditional machine learning algorithms often focus on the abstract model learning part. A traditional machine learning algorithm usually uses pre-extracted feature vectors as its input, which rarely pays attention to data acquisition. Instead, ML algorithms assume the feature vectors satisfy some mathematical or statistical properties and constraints, and learn machine learning models based on the feature vectors and their assumptions.

An important portion of machine learning research is focused on the theoretical guarantees of machine learning algorithms. For example, under certain assumptions on the feature vectors, what is the upper or lower bound of the accuracy *any* machine learning algorithm can attain? Such theoretical studies are sometimes not considered as a topic of PR research. Pattern recognition research and practice often has a stronger system flavor than that in machine learning.

We can find other differences between PR and ML. But, although we do not agree on expressions like "PR is a branch of ML" or "ML is a branch of PR," we do not want to emphasize the differences either.

PR and ML are two closely related subjects, and the differences may gradually disappear. For example, the recent deep learning trend in machine learning emphasizes *end-to-end* learning: the input of a deep learning method is the raw input data (rather than feature vectors), and its output is the desired prediction.

Hence, instead of emphasizing the differences between PR and ML, it is better to focus on the important task: let us just solve the problem or task that is presented to us!

One more note: the patterns or regularities recognized by PR research and practice involve various types of sensory input data, which means that PR is also closely related to subjects such as computer vision, acoustics, and speech processing.

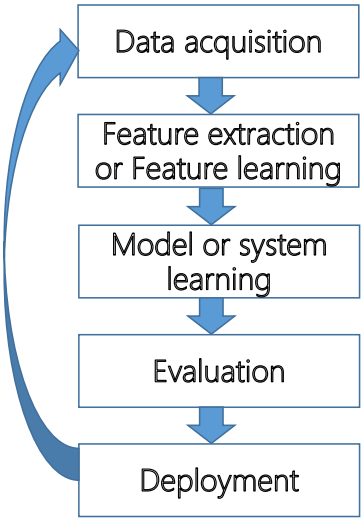


Figure 3: A typical pattern recognition pipeline with feedback loop.

2.3 Evaluation, deployment, and refinement

The next step after obtaining a model is to apply and evaluate the model. Evaluation of a PR or ML method or model is a complex issue. Depending on the project goals, various performance measures such as accuracy (or error) rate, speed, resource consumption, and even R&D costs must be taken into account in the evaluation process.

We will leave the discussions on evaluation to Chapter 4. Suppose a PR model or system has passed the evaluation process (i.e., all the design targets have been met by the system); the next step is to deploy the system into its real-world application environments. However, passing evaluations and tests in a laboratory setting does not necessarily mean the PR system works well in practice. In fact, in many if not most cases, the reverse is true. The deployment may encounter environments that are far more complex than what are expected or assumed during the research, development, and evaluation phases. The real-world raw input data may, not surprisingly, have different characteristics than the data collected for training purposes.

Hence, deployment is rarely the last step in a PR system's life cycle. As shown in Figure 3, all the issues that appear in the system deployment phase have to be carefully collected, studied, and corrected (as reflected by the arrow pointing from "Deployment" to "Data acquisition"). The pipeline in Figure 3 now supersedes that in Figure 1 in order to reflect the importance of feedback.

Depending on the issues, some or even all the steps (data acquisition, feature extraction or learning, model learning and evaluation) may have to be refined or completely revamped. This refinement process may require many cycles of effort. Of course, if issues are identified earlier in the evaluation step, the system

has to be revised before its deployment.

Both data acquisition and manual feature extraction based on knowledge from domain experts are application or domain specific, which will involve background from various subject areas. In this book, we will not be able to go into details of these two steps.

The other steps are introduced in this book. For a detailed description of the structure and contents of the rest of this book, please refer to the preface.

Exercises

1. Below is an interesting equation I saw from a short online entertainment video clip:

$$\sqrt[3]{a+\frac{a+1}{3}\sqrt{\frac{8a-1}{3}}}+\sqrt[3]{a-\frac{a+1}{3}\sqrt{\frac{8a-1}{3}}}\,. \tag{1}$$

What do you think this equation will evaluate to? Note that we only consider real numbers (i.e., complex numbers shall not appear in this problem).

One does not often see this kind of complex equation in an online entertainment video, and this equation almost surely has nothing to do with pattern recognition or machine learning. However, as will be illustrated in this problem, we can observe many useful thinking patterns in the solution of this equation that are also critical in the study of machine learning and pattern recognition. So, let us take a closer look at this equation.

(a) **Requirements on the input.** In a pattern recognition or machine learning problem, we must enforce some constraints on the input data. These constraints might be implemented by *pre-processing* techniques, by requirements in the *data acquisition* process, or by other means.

The requirement for the above equation is specified in terms of a. What shall we enforce on the variable a?

(b) **Observing the data and the problem.** The first step in solving a PR or ML problem is often *observing* or *visualizing* your data—in other words, to gain some intuitions about the problem at hand. While trying to observe or visualize the data, two kinds of data are often popular choices: those data that are *representative* (to observe some common properties), and those that have *peculiar* properties (to observe some corner cases).

One example of peculiar data for Equation 1 is $a = \frac{1}{8}$. This value for a is peculiar because it greatly *simplifies* the equation. What is Equation 1's value under this assignment of a?

(c) **Coming up with your idea.** After observing the data, you may come up with some intuitions or ideas on how to solve the problem. If that idea is *reasonable*, it is worth pursuing.

Can you find another peculiar value for a? What is Equation 1's value in that case? Given these observations, what is your idea about Equation 1?

(d) **Sanity check of your idea.** How do you make sure your idea is reasonable? One commonly used method is to test it on some simple cases, or to write a simple prototype system to verify it.

For Equation 1, we can write a single-line Matlab/Octave command to evaluate its value. For example, let `a=3/4` assigns a value to a; we can evaluate Equation 1 as

```
f = ( a + (a+1)/3*sqrt((8*a-1)/3) )^(1/3) + ...
    ( a - (a+1)/3*sqrt((8*a-1)/3) )^(1/3)
```

What is the value this command returns?

(e) **Avoiding caveats in coding.** The value returned by this command is obviously wrong—we know the result must be a real number. What is the cause of this issue? It is caused by a small caveat in the programming. We should pay special attention to programming details in our prototype system, in order to make sure it correctly implements our idea.

Read the online Matlab manual and try to fix the problem. What is the correct value? If you use the correct code to evaluate Equation 1 for many different a values ($a \geq 0.125$), can it support your idea?

You might have come up with a good idea that is supported by your code from the very beginning. In this case, you can move to the next part. If not, please observe the data, come up with an idea that is better than your original one, and test it until it is supported by your sanity check experiments.

(f) **Formal and rigorous proof.** Inevitably you will have to formally prove your statements in some tasks. A proof needs to be correct and rigorous. The first step in a valid proof is probably to define your *symbols and notations* such that you can *express your problem and idea precisely* in mathematical language.

Define your notations and write down your idea as a precise mathematical statement. Then, prove it rigorously.

(g) **Making good use of existing results when they are available.** In both research and development, we have to make good use of existing resources—e.g., mathematical theorems, optimization methods, software libraries, and development frameworks. That said, to use existing results, resources, and tools means that you have to be aware of them. Thus, it is good to *know major results and tools in subject domains that are close to one's own domain*, even if you are not familiar with their details.

Of course, these resources and tools include those that are developed by yourself. Use the theorem you just proved in this problem to calculate the following expression:

$$\sqrt[3]{2+\sqrt{5}} + \sqrt[3]{2-\sqrt{5}}\,.$$

(h) **Possibly extending your results to a more general theory.** Some of your results may have the potential to become a theory that is more general and more useful. And, it is worthwhile to take this step when there is an indication of such a possibility.

The above equation in fact comes from a more general result: Cardano's method to solve a cubic equation. Gerolamo Cardano was an Italian mathematician, and he showed that the roots of the equation

$$z^3 + pz + q = 0$$

can be solved using expressions related to Equation 1. Read the information at https://en.wikipedia.org/wiki/Cubic_equation (especially the part that is related to Cardano's method), and try to understand this connection.