Introduction

Jianxin Wu

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

April 20, 2017

Contents

This course introduces pattern recognition, with a sizable portion of its contents being the introduction to various machine learning algorithms.

What is pattern recognition, and what is machine learning? Pattern recognition is introduced as follows in Wikipedia as of Oct 8, 2016, which also involves machine learning.

> 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 inside ML; ML is* not *a branch of PR either*.

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 commercially available as of today) may operate as follows.

`T.1` The car identifies its owner is approaching and automatically unlock 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 the above task `T.1`, 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., detection and recognition of objects (such as identifying a person) and identifying of anomalies (such as finding 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 image or video understanding results, 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, human identity recognition (such as face recognition based on human faces, or gait recognition based on a person's walking style and habits).

The 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 finish 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 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. Techniques from various areas are required to finish 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, understanding their meanings and choosing appropriate answers through natural language processing, and speaking its answer back through speech synthesis.

`T.2` involves two closely related subjects: speech processing and natural language processing. The input of `T.2` are speech signals obtained via one or several microphones, which will undergo several layers of processing: the microphones' electronic signal is first converted into meaningful words, phrases or sentences by speech recognition; the natural language processing module will convert these words into representations such 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 shall convert the text sentences into sound signal, which will be spoken to the user 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.

We will leave the analyses of `T.3` and `T.4` as exercises.

In this example, we have witnessed many sensors (e.g., camera, infrared camera, microphone) and outputs of various modules (e.g., existence of 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 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 avoid collision if obstacles do exist.

2 PR & ML

The above examples are all examples of *pattern recognition*, which automatically extracts useful patterns from input data (e.g., pedestrian 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 finding appropriate response to user's speech

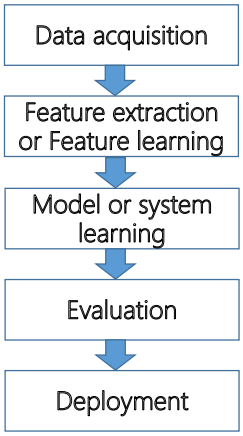


Figure 1: A typical pattern recognition pipeline.

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 is in general automatic (i.e., without human in the loop).

2.1 A typical PR pipeline

We list the steps in a typical pattern recognition (PR) pipeline in Figure 1.

A PR pipeline often starts from its input data, which most probably comes from various sensors. Sensors (such as cameras and microphones) collect input from the environment in which the PR system operates. This step, also termed as *data acquisition*, is extremely important for pattern recognition performance. The properties of input data can be more important than other steps or components. Face recognition from surveillance videos is such an example. 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., longer than 30 meters), a face will occupy only less than 20×20 pixels in the video frames. This resolution is too small to provide useful identity information for a suspect, no matter for a human 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.

Acquiring high quality sensory input involves experts from many subjects, for example, physicists, acoustic engineers, electrical and electronics engineers, optical engineers. Sensory input data often need to be digitized, which may

appear in many different forms such as texts, 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 as its input.

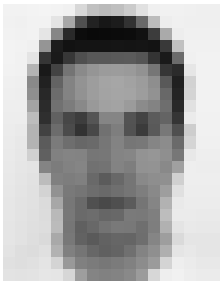
The next step is feature extraction or feature learning. The raw sensory input, even after digitization, are often far from meaningful or interpretable. For example, a color picture with resolution 1024×768 and three channels is digitized as $3 \times 1024 \times 768 = 2,359,296$ integers between 0 and 255. These large quantity of integers are not very useful for finding useful pattern or regularity in the image. Figure 2 shows a small gray-scale (single channel) face image (2a) and its raw input format (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 impossible 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., turn these 414 numbers into other numerical values which are useful for finding faces. For example, because most eyes are darker than other face 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 image (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. Of course, this single feature is very weak and 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 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 these 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 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 algorithms.

After the feature extraction or learning step, we need to produce a model, which takes the feature vectors as its input, and produce 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 $\boldsymbol{w} \in \mathbb{R}^d$ and $b \in \mathbb{R}$ are $d + 1$ *parameters* of the linear machine learning



(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 6 times) in (2a). It is seen by the computer as a 23×18 matrix of integers, as shown in (2b).

model. Given any image with a feature vector $\boldsymbol{x}$, the model will predict the image as a face image if $\boldsymbol{w}^T\boldsymbol{x} + b \geq 0$, and non-face if $\boldsymbol{w}^T\boldsymbol{x} + b < 0$. In this parametric form of machine learning 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., use experiences (training examples and their labels) to learn a model, which can predict for future examples which are not observed during the learning process.

2.2 PR vs. ML

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, while feature learning (also called

representation learning) has increasingly attracted more attention in the ML community.

However, PR include more than the components that are ML-related. Data acquisition, 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 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 often focus on the abstract model learning part. A traditional machine learning algorithm usually use 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. A large portion of machine learning researches are on theoretical guarantees of the 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 not considered as the topic in PR research. Pattern recognition researches and practices often have a stronger system flavor than that in machine learning.

We may find more differences between PR and ML. But, although we do not agree on expressions like "PR is a branch in ML" or "ML is a branch in 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: solve the problem (should it be a practical or a theoretical problem).

The pattern or regularity recognized by PR researches and practices involve different sensory input data, which means that PR is also closely related to subjects such as computer vision, acoustics and speech.

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.

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 the

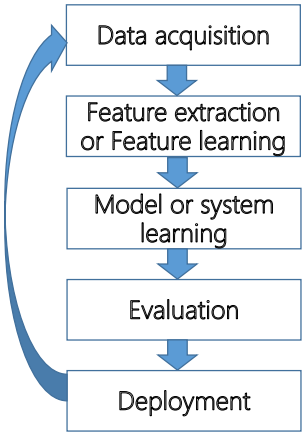


Figure 3: A typical pattern recognition pipeline with the feedback loop.

lab evaluations and tests does not necessarily mean the PR system works well in practice. The deployment may encounter environments that are 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").

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

3 Structure of this course

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

The other steps are introduced in this course, and the rest of this course is organized as follows.

Part 1 Introduction and preliminaries. This part introduces the basic concepts of pattern recognition and machine learning, the mathematical background that are required for this course, and use an example to introduce various components in a typical pattern recognition pipeline.

(a) This note introduces the basic concepts of PR and ML, and briefly introduces the connections and differences between these two subjects.

(b) The second note introduces preliminaries for this course, including mainly linear algebra, basic probability and mathematical statistics, and a brief introduction of optimization and matrix calculus. The purpose of this note is to make this course self-contained.

(c) The third note first introduces a pattern recognition example: face recognition. A simple nearest neighbor classifier is introduced to provide a (far from satisfactory) solution to this task. The nearest neighbor classifier, although extremely simple, possess all necessary components of a PR and ML system. We use it as example to introduce various system steps, especially the factors (details) that make PR and ML difficult. We also introduce a framework that is useful in formulating and solving PR and ML tasks.

(d) The fourth note introduces the evaluation step, assuming a method or system have been developed. We first introduce accuracy and error, two commonly used metrics in evaluation, and the overfitting and underfitting concepts. With the help of these definitions and concepts, we transform the framework in the third note into a formal cost (or risk) minimization framework, which extends the error rate into a more general cost concept. This note also introduces evaluation metrics for more complex scenarios. A brief introduction of the Bayes error rate is discussed to setup the lower bound of the error rate for a specific problem (or equivalently, how accurate a system can be?). Finally, this note ends with a brief discussion of statistical tests, which tells us how confident we can put in our evaluation results.

Part 2 Domain independent feature extraction. Although we will not touch domain-dependent feature extraction, there are some feature extraction techniques that are suitable across different task domains. We introduce two such techniques in this part.

(a) The fifth note introduces the principal component analysis (PCA), which is an unsupervised feature dimensionality reduction method. PCA extracts new features from the input feature vector without using any label associated with these feature vectors.

(b) The sixth note discusses the Fisher's Linear Discriminant (FLD), a supervised feature dimensionality reduction method. By using labels accompanying the feature vectors, FLD is able to extract features that are more powerful than PCA.

Part 3 Three classic classifiers. Classification is the most studied topic in machine learning and pattern recognition. We discuss three types of classifiers that are widely used in both PR and ML.

(a) The seventh note introduces the support vector machine (SVM), which is a classic classification method. SVM can have linear or non-linear classification boundaries, can handle tasks with two or more classes, and can handle tasks that have zero or non-zero training errors. The focus of this note is the ideas behind SVM and how these ideas are beautifully formalized and solved as mathematical optimization problems. Another key point of this note is to study how SVM starts from the simplest case and gradually changes itself to solve the most general and complex case.

(b) The eighth note discusses classifiers designed explicitly in the probabilistic point of view. We start from a few concepts that are easily confused, and then the Bayes' theorem, cornerstone of probabilistic classifiers. Concepts such as generative and discriminative models, and parametric and nonparametric models are discussed, followed by parameter estimation in both parametric and nonparametric cases.

(c) The ninth note is on metric learning and data normalization. Comparing the similarity or dissimilarity of two feature vectors are vital in many machine learning methods. We first define distance metrics, then introduce a few widely used distance metrics. In this note, we also discuss data transformation and normalization. We discuss a few techniques for data transformation and normalization, using linear regression as an example.[1]

(d) The tenth note introduces decision tree, a classic classifier that is particularly useful in handling nominal features. We only provide a simple decision tree classifier using information gain as its tree building tool. Information gain is a concept proposed in the information theory. Information theory was proposed in the communication area, but many of its concepts are useful tools for PR and ML. In this note, we provide a brief introduction to basic information theory concepts and tools, such as entropy, mutual information and relative entropy.

Part 4 Data with special characteristics and their handling. The world is complex, which produce complex data (or feature vectors). In part 3, we have assumed that dimensions in the data (or feature vectors) are independent of each other. This assumption is easily violated in real world applications.

(a) The eleventh note is on sparse and misaligned data. Two types of special data are discussed in this note. Some data exhibit inherent sparsity. In this note, we introduce some basic concepts in sparse learning methods. The other type of data discussed in this note is misaligned sequential data. We introduce dynamic time warping (DTW) to handle such misaligned data. DTW is solved via dynamic programming.

[1] Regression is an important topic in machine learning too. Although we will not discuss regression in detail, this note provides a peek on regression.

(b) The twelfth note is on the hidden Markov model (HMM), a classic tool to handle sequential data. We start this note by introducing a few types of sequential data, the Markov property and the hidden Markov model. Three key problems are then identified in learning HMM. The rest of this note is devoted to solving these three key problems. Dynamic programming turns out to be the key idea behind many algorithms introduced in this note.

Part 5 Advanced topics. The last part in this course introduces three advanced topics, which are beyond the scope of an undergraduate introductory course. However, these materials are useful for students that are interested in advancing their studies in PR and ML. These materials are optional.

(a) The thirteenth note describes details of the normal distribution. Normal distribution, also called the Gaussian distribution, is the most widely used continuous probability distribution in probabilistic methods. Beyond introducing the properties of normal distributions, we also introduce two applications of these properties: in parameter estimation and in the Kalman filtering.

(b) The fourteenth note is on the expectation-maximization (EM) algorithm for parameter estimation in probabilistic models. We use the Gaussian mixture model (GMM) as an example to introduce the EM algorithm.

(c) The fifteenth (and last) note describes the convolutional neural network (CNN), a typical deep learning model which has shown excellent accuracy in dealing with images and videos.

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 always see this kind of complex equations in an entertainment online 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, which 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 or 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 intuition 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 you 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? Commonly used methods include 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 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 come to the next part. If not, please observe the data, come up with a better idea 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 the mathematical language.

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

(g) **Make 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 mean 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 extend 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 do so when there is sign indicating such a possibility.

The above equation in fact comes from a more general result: Cardano's method to solve a cubic function. Gerolamo Cardano is 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_function (especially the part that is related to Cardano's method), and try to understand this connection.