So, How do You Define a Machine Learning Problem?

The first step in any project is defining your problem. You can use the most powerful and shiniest algorithms available, but the results will be meaningless if you are solving the wrong problem.

Thinking deeply about your problem before you get started is unarguably the most important aspect of applying machine learning.

We can define a Problem Definition Framework in order to reach a viable outcome at the end of our venture and make some decisions (hopefully right decisions) to determine what's the best thing we can do to develop an acceptable solution for that particular problem.

Note: Sometimes the problem statement is not necessarily vague or complicated. In contrast, sometimes they are very straightforward. You have to decide if it's necessary to go through all these steps. If not, you can get down to business right off the bat.

The Problem Definition Framework

A) Is there a pattern?

The fundamental assumption that underlies all machine learning problems is that there is a pattern. So before you start, ask yourself, do you think there's a pattern. If not, then we are already done.

That pattern we're looking for is some function *f*, which maps some input *X* onto an output *Y i.e,* $f : y(x)$. Of course, we don't know *f – This is the entire crux of machine learning: if I have some pattern that I can't directly observe, how can I at least approximate it?*

B) Informal description

Describe the problem as though you were describing it to a friend or colleague. This can provide a great starting point for highlighting areas that you might need to fill. It also provides the basis for a one sentence description you can use to share your understanding of the problem.

For example: I need a program that will tell me which tweets are likely to get retweets.

Or, say, We're building an application to help people keep their photos organized. Our target user base are casual smartphone photographers who have a large number of photos in their camera roll. We want to make it easier for these users to find photos of interest.

Notice how vague that problem statement is - what defines a photo of interest? There's a multitude of ways we could address this task and without understanding the problem in more detail we won't know which direction to take. At this point in time, we have insufficient information to specify an objective function for training a model.

Understanding the problem and developing the requirements isn't something you typically get right on the first attempt; this is often an *iterative process* where we initially define a set of coarse requirements and refine the detail as we gain more information.

C) Why does this problem needs to be solved?

— What's the motivation: Consider your motivation for solving the problem. What need will be fulfilled when the problem is solved? What is it you're out for?

For example, you may be solving the problem as a learning exercise. This is useful to clarify as you can decide that you don’t want to use the most suitable method to solve the problem, but instead you want to explore methods that you are not familiar with in order to learn new skills.

Alternatively, you may need to solve the problem as part of a duty at work, ultimately to keep your job.

— Solution Benefits: Consider the benefits of having the problem solved. What capabilities does it enable?

It is important to be clear on the benefits of the problem being solved to ensure that you capitalize on them. These benefits can be used to sell the project to colleagues and management to get buy in and additional time or budget resources.

If it benefits you personally, then be clear on what those benefits are and how you will know when you have got them. For example, if it’s a tool or utility, then what will you be able to do with that utility that you can’t do now and why is that meaningful to you?

— Solution Use: Consider how the solution to the problem will be used and what type of lifetime you expect the solution to have.

As programmers we often think the work is done as soon as the program is written, but really the project is just beginning it’s maintenance lifetime. The way the solution will be used will influence the nature and requirements of the solution you adopt.

Consider whether you are looking to write a report to present results or you want to operationalize the solution. If you want to operationalize the solution, consider the functional and non-functional
requirements you have for a solution, just like a software project.

D) Make assumptions

Create a list of assumptions about the problem and it’s phrasing. These may be rules of thumb and domain specific information that you think will get you to a viable solution faster.

Observe your data (if there's any) and make a list of possible reasons that may explain what's going on or how the data is related to your particular problem. You should also make a list of points that you think is relative and will help you better understand the problem. This way you will know if you have to collect more data or how much data should suffice to make reasonable predictions.

The problem is that even with those observations, an infinite number of possibilities could explain the patterns in the data. We’re trying to pluck some hypothesis (g) that we think works best from an infinite space of different hypotheses (H). There's no reason to believe, we will find the best solution right off the bat. But, don't worry. The algorithm will be responsible for picking through our infinitely large hypothesis space, H and figuring out which one makes the most sense.

These hypotheses or assumptions can also be useful to highlight areas of the problem specification that may need to be challenged, relaxed or tightened. After all, breakthroughs and innovations occur when assumptions and best practices are demonstrated to be wrong in the face of real data.

E) Understand the problem from the perspective of the user

If you want to solve a machine learning problem successfully and build a QuAM that performs efficiently the best thing you can do is pinpoint the features that your dataset should be comprised of. You have to have a clear view of your goal. And what better way to know what problem(s) to solve than asking the users themselves what they want. No one knows the problem better than the intended user - the person we are attempting to solve the problem for.

Perform informational interviews with end users to understand their perspective. The further removed you are from the end user, the less likely you are to solve the actual problem they're experiencing.

F) Subject yourself to the problem

Pay close attention to how you solve the task, as this might inform what features might be important to include when you do train a model to perform the task.

Explore how you would solve the problem manually. List out step-by-step what data you would collect, how you would prepare it and how you would design a program to solve the problem. This may include prototypes and experiments you would need to perform which are a gold mine because they will highlight questions and uncertainties you have about the domain that could be explored.

This is a powerful tool. It can highlight problems that actually can be solved satisfactorily using a manually implemented solution. It also flushes out important domain knowledge that has been trapped up until now like where the data is actually stored, what types of features would
be useful and many other details.

Collect all of these details as they occur to you and update the previous sections of the problem definition. Especially the assumptions and rules of thumb.

▼ Related Blogs & Articles

- https://machinelearningmastery.com/how-to-define-your-machine-learning-problem/
- https://www.kdnuggets.com/2018/10/define-machine-learning-problem-detective.html
- https://www.jeremyjordan.me/ml-requirements/