**How to Define Your 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.

In this post you will learn the process for thinking deeply about your problem before you get started. This is unarguably the most important aspect of applying machine learning.

**Problem Definition Framework**

I use a simple framework when defining a new problem to address with machine learning. The framework helps me to quickly understand the elements and motivation for the problem and whether machine learning is suitable or not.

The framework involves answering three questions to varying degrees of thoroughness:

Step 1: What is the problem?

Step 2: Why does the problem need to be solved?

Step 3: How would I solve the problem?

**Step 1: What is the Problem**

The first step is defining the problem. I use a number of tactics to collect this information.

**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 will get retweets.*

**Formalism**

In a previous blog post [defining machine learning](http://machinelearningmastery.com/what-is-machine-learning/) you learned about Tom Mitchell’s machine learning formalism. Here it is again to refresh your memory.

*A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E.*

Use this formalism to define the *T*, *P*, and *E* for your problem.

For example:

Task (*T*): Classify a tweet that has not been published as going to get retweets or not.

Experience (*E*): A corpus of tweets for an account where some have retweets and some do not.

Performance (*P*): Classification accuracy, the number of tweets predicted correctly out of all tweets considered as a percentage.

**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.

It can be useful to highlight questions that can be tested against real data because breakthroughs and innovation occur when assumptions and best practice are demonstrated to be wrong in the face of real data. It can also be useful to highlight areas of the problem specification that may need to be challenged, relaxed or tightened.

For example:

The specific words used in the tweet matter to the model.

The specific user that retweets does not matter to the model.

The number of retweets may matter to the model.

Older tweets are less predictive than more recent tweets.

**Similar problems**

What other problems have you seen or can you think of that are like the problem you are trying to solve? Other problems can inform the problem you are trying to solve by highlighting limitations in your phrasing of the problem such as time dimensions and conceptual drift (where the concept being modeled changes over time). Other problems can also point to algorithms and data transformations that could be adopted to spot check performance.

For example: A related problem would be email spam discrimination that uses text messages as input data and needs binary classification decision.

**Step 2: Why does the the problem need to be solved?**

The second step is to think deeply about why you want or need the problem solved.

**Motivation**

Consider your motivation for solving the problem. What need will be fulfilled when the problem is solved?

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 nonfunctional requirements you have for a solution, just like a software project.

**Step 3: How would I solve the problem?**

In this third and final step of the problem definition, 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.

We have considered a manually specified solution before when describing complex problems in [why machine learning matters](http://machinelearningmastery.com/machine-learning-matters/).

**Summary**

In this post you learned the value of being clear on the problem you are solving. You discovered a three step framework for defining your problem with practical tactics at at step:

Step 1: What is the problem? Describe the problem informally and formally and list assumptions and similar problems.

Step 2: Why does the problem need to be solve? List your motivation for solving the problem, the benefits a solution provides and how the solution will be used.

Step 3: How would I solve the problem? Describe how the problem would be solved manually to flush domain knowledge.

How do you define your problem for machine learning? Have you used any of the above tactics and if so, what were your experiences? Leave a comment.