Identifying Good Problems for ML

This section examines the characteristics of good ML problems.

Clear Use Case

ith the problem, **not** the solution. Make sure you aren't treating ML as a hammer for your problems.

Focus on problems that would be difficult to solve with traditional programming. For example, consider Smart Reply (https://research.googleblog.com/2017/05/efficient-smart-reply-now-for-gmail.html). The Smart Reply team recognized that users spend a lot of time replying to emails and messages; a product that can predict likely responses can save user time. Another example is in Google Photos, where the business problem was to find a specific photo by keyword search without manual tagging.

Imagine trying to create a system like Smart Reply or Google Photos search with conventional programming. There isn't a clear approach. By contrast, machine learning can solve these problems by examining patterns in data and adapting with them. Think of ML as just one of the tools in your toolkit and only bring it out when appropriate.

With these examples in mind ask yourself the following questions:

1. What problem is my product facing?
2. Would it be a good problem for ML?

Don't ask the questions the other way around!

Know the Problem Before Focusing on the Data

pared to have your assumptions challenged.

If you understand the problem clearly, you should be able to list some potential solutions to test in order to generate the best model. Understand that you will likely have to try out a few solutions before you land on a good working model.

Exploratory data analysis can help you understand your data, but you can't yet claim that patterns you find generalize until you check those patterns against previously unseen data. Failure to check could lead you in the wrong direction or reinforce stereotypes or bias.

Lean on Your Team's Logs

ɿuires a lot of relevant data.

Data collected specifically for your task is going to be the most useful. In practice, you may not be able to do this, and you'll rely on whatever data you can get that's close enough. That's fine as long as you're aware of the cost, and as you can eventually get product logs, you can use those to build something more targeted to your task.

How much is "a lot?" That depends on the problem, but more data typically improves your model and therefore your model's predictive power. A good rule of thumb is to have at least thousands of examples for basic linear models, and hundreds of thousands for neural networks. If you have less data, consider a non-ML solution first.

Predictive Power

ɜatures contain predictive power.

Suppose you're trying to predict which horses will perform well in a race. You decide to tackle the problem with ML and use the horse’s eye color as a feature. You reason that eye color predicts which horses are prone to eye disease, which in turn could predict a horse’s speed and stamina. Maybe you’re wrong and you'll reject the hypothesis later based on evidence; that is, perhaps using eye color as a feature does not improve your model.

You should not try to make ML do the hard work of discovering which features are relevant for you. If you simply throw everything at the model and see what looks useful, your model will likely wind up overly complicated, expensive, and filled with unimportant features. In smaller datasets, you have a higher chance that a feature will be correlated with your label by chance within your sample of data. If you try lots of features without a hypothesis, you'll falsely believe these are relevant signals for your model. You wouldn't catch this until you tried to make predictions with your model and realized it did not generalize.

Predictions vs. Decisions

make decisions, not just predictions.

By *decisions,* we mean that your product should take action on the output of the model. ML is better at making decisions than giving you insights. If you have a bunch of data and want to find out "interesting" things about it, statistical approaches make more sense.

Make sure your predictions allow you to take a useful action. For example, a model that predicts the likelihood of clicking certain videos could allow a system to prefetch the videos most likely to be clicked.

Sometimes the prediction and decision are closely aligned, but in other cases, the relationship is less apparent. Review the table below for examples of some prediction/decision pairs.

Prediction	Decision
What video the learner wants to watch next.	Show those videos in the recommendation bar.
Probability someone will click on a search result.	If P(click) > 0.12, prefetch the web page.
What fraction of a video ad the user will watch.	If a small fraction, don't show the user the ad.

Previous
← The ML Mindset (/machine-learning/problem-framing/big-questions)

Next
Hard ML Problems (/machine-learning/problem-framing/hard) →

Last updated 2019-11-04 UTC.