Basic Error Analysis

13

Error analysis: Look at dev set examples to evaluate ideas

When you play with your cat app, you notice several examples where it mistakes dogs for cats. Some dogs do look like cats!

A team member proposes incorporating 3rd party software that will make the system do better on dogs. These changes will take a month, and the team member is enthusiastic. Should you ask them to go ahead?

Before investing a month on this task, I recommend that you first estimate how much it will actually improve the system's accuracy. Then you can more rationally decide if this is worth the month of development time, or if you'd be better off using that time on other tasks.

In detail, here's what you can do:

1. Get a sample of (say) 100 dev set examples that your system *misclassified*. I.e., examples that your system made an error on.
2. Look at these examples manually, and count what fraction of them are dog images.

The process of looking at misclassified examples is called "error analysis." In this example, if you find that only 5% of the misclassified images are dogs, then no matter how much improvement you make to the dog problem, you won't get rid of more than 5% of your errors. In other words, 5% is a "ceiling" (meaning maximum possible amount) that the proposed project could help. Thus, if your overall system is currently 90% accurate (10% error), this improvement is likely to result in at best 90.5% accuracy (or 9.5% error, which is 5% less error than the original 10% error).

In contrast, if you find that 50% of the mistakes are dogs, then you can be more confident that the proposed project can have a big impact. It could boost accuracy from 90% to 95% (a 50% relative reduction in error, from 10% down to 5%).

This simple counting procedure of error analysis gives you a quick way to estimate the possible value of incorporating the 3rd party software to do better on dogs. It provides a quantitative basis on which to decide whether to make this investment.

Error analysis can often help you figure out how promising different directions are. I've seen many engineers reluctant to carry out error analysis. It often feels more exciting to just jump in and implement some idea, rather than question if the idea is worth the time investment. This is a common mistake: It might result in your team spending a month only to realize afterward that it resulted in little benefit.

Manually examining 100 examples does not take long. Even if you take one minute per image, you'd be done in under two hours. These two hours could well save you a month of wasted effort, and is time well spent.

"Error Analysis" refers to the process of examining dev set examples that your algorithm misclassified, so as to understand the underlying causes of the errors. This can both help you prioritize projects—as in this example—and inspire new directions, which we will discuss next. The next few chapters will also present best practices for carrying out error analyses.