Error Analysis

There is an old saying that you learn more from your mistakes than from your successes.

This is true in training models as well.

Hyper-focus on the Performance Metric. which is a *summary* statistic, may mask subtle problems in the model.

By examining where the model succeeds and where it fails

- We may be able to improve the Performance Metric by adjusting the model to make correct predictions on previously incorrect examples
- Uncover undesirable properties of the model: bad predictions that are associated with a particular subset of examples
    - so overall Performance Metric may be good, but *always* wrong on important examples

The main message is that, to improve models, we must go deeper than summary statistics, such as the Performance Metric or Average Loss.

- We must examine errors on validation examples
    - To determine if the errors are systematic
    - Find ways to correct

Perhaps having answers to the following questions is *the most important* topic from this course

- How can you *improve* your models ?
- How can you tell if your model's predictions "make sense"
- How can I be most productive ?

Introduction

We glossed over Error Analysis in our first pass at the Classification task.

We motivate the need for Error Analysis using the MNIST digits classification task.

Let's visit the notebook [Introduction](Error_Analysis.ipynb#Classification:-Beyond-accuracy)

Classification: Error Analysis

As we did with the Classification task

- We will start with Error Analysis for Binary Classification
- Move onto Error Analysis for Multinomial Classification

In addition to the all-encompassing Accuracy measure, we will examine several measures of Conditional Accuracy.

Let's visit the notebook section [Conditional Accuracy (Error_Analysis.ipynb#Binary-classification:-Conditional-accuracy)](Error_Analysis.ipynb#Binary-classification:-Conditional-accuracy)

Balancing Precision and Recall

It turns out that there are competing goals for Classification that are hard to jointly satisfy:

- Correctly identify all Positive examples in a set (Recall)
- Don't mis-identify an example as Positive unless it really is Positive (Precision)

To see how they compete, consider

- Suppose we predict *all* examples to be Positive
    - We get perfect Recall, potentially by reducing Precision.
- Suppose we only predict Positive on the *single* example of which we are most certain
    - If we're right, we get perfect Precision, but with poor Recall.

Let's return to the notebook section [Precision versus Recall (Error_Analysis.ipynb#Precison/Recall-Tradeoff)](Error_Analysis.ipynb#Precison/Recall-Tradeoff) to explore the trade-off.

Multinomial Classification: Error Analysis

Conditional Accuracy can be generalized to the Multinomial Classification task.

Let's return to the notebook section Multinomial Classification (Error_Analysis.ipynb#Multinomial-classification:-Confusion-matrix)

Error Analysis for the Regression task

We have thus far focused on Error Analysis for the Classification task.

Let's return to the notebook section on Error Analysis for Regression (Error_Analysis.ipynb#Regression:-beyond-RMSE/R^2) to discuss the analysis of errors for a Regression task.

Hands-on: Beneath the covers of a classifier for MNIST digits

Time to roll up your sleeves and get your hands dirty !

We will dig into the results a classifier for MNIST digit recognition.

Let's prepare to code as we visit [the notebook (Error_Analysis_MNIST.ipynb)]() that performs Error Analysis on an MNIST digit classifier.

Summary

Our goals in this presentation were

- To emphasize **not stopping** your exploration once a summary statistic is "good enough"
    - Explore beneath the surface for systematic errors
- To emphasize the rule of "no magic"
    - Your eyes and your intuition are valuable tools, *when coupled with*
        - Knowledge of the underlying math and algorithms
        - The will to write a little code

Becoming a more successful Data Scientist often comes down to using the concepts and tools of Error Analysis.

In [3]:

```
print("Done")
```

Done